Cat. No.
I52E-DE-03

OMRON

Trajexia Bewegungssteuerungssystem

TJ1-MC04
TJ1-MC16

PROGRAMMIERHANDBUCH

Hinweis

OMRON-Produkte sind zum Gebrauch durch einen qualifizierten Bediener gemäß angemessenen Verfahren und nur zu den in diesem Handbuch beschriebenen Zwecken gefertigt.
In diesem Handbuch werden Sicherheitshinweise nach folgenden Konventionen gekennzeichnet und eingeteilt. Beachten Sie stets die in diesen Hinweisen enthaltenen Informationen. Ein Nichtbeachten der Sicherheitshinweise kann zu Verletzungen oder zu Sachschäden führen.

Definition von Sicherheitsinformationen

GEFAHR

Kennzeichnet eine bevorstehende gefährliche Situation, die zum Tod oder zu ernsthaften Verletzungen führt, wenn sie nicht vermieden wird.

VORSICHT

Kennzeichnet eine potenziell gefährliche Situation, die zum Tod oder zu schweren Verletzungen führen kann, wenn sie nicht vermieden wird.

Achtung

Kennzeichnet eine potenziell gefährliche Situation, die zu kleineren oder mittelschweren Verletzungen oder Sachschäden führen kann, wenn sie nicht vermieden wird.

Warenzeichen und Urheberrechte

PROFIBUS ist ein eingetragenes Warenzeichen von PROFIBUS International.
MECHATROLINK ist ein eingetragenes Warenzeichen der Yaskawa Corporation.
DeviceNet ist ein eingetragenes Warenzeichen der Open DeviceNet Vendor Assoc INC.
CIP ist ein eingetragenes Warenzeichen der Open DeviceNet Vendor Assoc INC.
Trajexia ist ein eingetragenes Warenzeichen von OMRON.
Motion Perfect ist ein eingetragenes Warenzeichen von Trio Motion Technology Ltd.

© OMRON, 2007

Alle Rechte vorbehalten. Diese Publikation darf ohne vorherige schriftliche Genehmigung von OMRON weder als Ganzes noch in Auszügen in irgendeiner Form oder auf irgendeine Weise, sei es auf mechanischem oder elektronischem Wege oder durch Fotokopieren oder Aufzeichnen, reproduziert, auf einem Datensystem gespeichert oder übertragen werden. In Bezug auf die hierin enthaltenen Informationen wird keine Patenthaftung übernommen. Da OMRON weiterhin an einer ständigen Verbesserung seiner Qualitätsprodukte arbeitet, sind Änderungen an den in diesem Handbuch enthaltenen Informationen ohne Ankündigung vorbehalten. Bei der Herstellung dieses Handbuchs wurden alle Vorsorgemaßnahmen ergriffen. Dennoch übernimmt OMRON keine Verantwortung für etwaige Fehler und Auslassungen. Ferner wird keine Haftung für Schäden übernommen, die durch die in diesem Handbuch enthaltenen Informationen entstehen.

Zu diesem Handbuch

Dieses Handbuch beschreibt die Installation und den Betrieb des Trajexia Motion Control Systems.
Lesen Sie dieses Handbuch und die anderen in der folgenden Liste aufgeführten Handbücher sorgfältig durch, und stellen Sie sicher, dass Sie die Informationen verstanden haben, bevor Sie versuchen, Trajexia Baugruppen zu installieren oder zu betreiben. Beachten Sie die Sicherheitshinweise im folgenden Abschnitt.

Bezeichnung	Kat.- Nr.	Inhalt
Trajexia Motion Controller Kurzanleitung	I50	Beschreibt, wie Sie sich schnell mit Trajexia vertraut machen; die Bewegung erfolgt in einem Test-Setup mit einer Achse über MECHATROLINK-II.
HARDWARE-REFERENZHANDBUCH für Trajexia Motion Controller	I51E	Beschreibt die Installation und Hardware-Spezifikationen der Trajexia-Baugruppen und erläutert die Trajexia Systemphilosophie.
Trajexia Motion Controller PROGRAMMIERHANDBUCH	I52	Beschreibt die für die Programmierung von Trajexia verwendeten BASIC-Befehle, erklärt die Kommunikationsprotokolle und die Trajexia Tools-Software, gibt praktische Beispiele und Informationen zur Fehlersuche.
Handbuch für Sigma-II-Servoantrieb	SIEP S800000 15	Beschreibt die Installation und den Betrieb von Sigma-II-Servoantrieben.
Handbuch für Servoantriebe der JUNMA -Serie	TOEP-C71080603 01-OY	Beschreibt die Installation und den Betrieb von JUNMA-Servoantrieben.
Handbuch JUSP-NS115	SIEP C71080001	Beschreibt die Installation und den Betrieb der MECHATROLINK-II-Optionskarte.
Handbuch für Sigma-III mit MECHATROLINK-Schnittstelle	SIEP S800000 11	Beschreibt die Installation und den Betrieb von Sigma-II-Servoantrieben mit MECHATROLINK-Schnittstelle.
V7-Frequenzumrichter	TOEP C71060605 02-OY	Beschreibt die Installation und den Betrieb von V7-Frequenzumrichtern.
F7Z-Frequenzumrichter	TOE S616-55 1-OY	Beschreibt die Installation und den Betrieb von F7Z-Frequenzumrichtern.
G7-Frequenzumrichter	TOE S616-60	Beschreibt die Installation und den Betrieb von G7-Frequenzumrichtern.
SI-T MECHATROLINK-Schnittstelle für G7 & F7	SIBP-C730600-08	Beschreibt die Installation und den Betrieb von MECHATROLINK-Schnittstellen für G7 und F7-Frequenzumrichter.
ST-T/V7 MECHATROLINK-Schnittstelle für V7	SIBP-C730600-03	Beschreibt die Installation und den Betrieb von MECHATROLINK-Schnittstellen für V7-Frequenzumrichter.
MECHATROLINK-E/A-Module	SIE-C887-5	Beschreibt die Installation und den Betrieb von MECHATROLINK-Eingangs- und Ausgangsmodulen und dem MECHATROLINK-II-Verstärker.
SYSMAC CS/CJ-Serie Kommunikationsbefehle	W342	Beschreibt das FINS-Kommunikationsprotokoll und die FINS-Befehle

VORSICHT
Fehler, die dadurch verursacht werden, dass Sie die in diesem Handbuch enthaltenen Informationen nicht lesen oder nicht verstehen, können zur Verletzung oder zum Tod von Personen, zur Beschädigung des Produkts bzw. zu Fehlfunktionen führen. Lesen Sie jedes Kapitel vollständig durch, und führen Sie die vorgestellten Maßnahmen und Bedienverfahren erst durch, wenn Sie sicher sind, dass Sie die im jeweiligen Kapitel und den zugehörigen Abschnitten bereitgestellten Informationen verstanden haben.

Unterstützung von Funktionalität nach Baugruppenversion

Während der Entwicklung von Trajexia wurde nach der Markteinführung neue Funktionalität in die Controller-Baugruppe übernommen.
Diese Funktionalität ist in die Firmware und/oder FPGA der Controller-Baugruppe implementiert.
In der unten stehenden Tabelle ist die entsprechende Funktionalität für die jeweilige Firmware- und FPGA-Version des TJ1-MC__ gezeigt.

Funktionalität	**TJ1-MC__ Firmware-Version**	**TJ1-MC__ FPGA-Version**
Volle Unterstützung für TJ1-FL02	V1.6509	21 und höher.
Unterstützung für BASIC-Befehle FINS_COMMS	V1.6509	Alle Ausführungen
Unterstützung für TJ1-DRT	V1.6509	Alle Ausführungen
Unterstützung für TJ1-MC04 und TJ1-ML04	V1.6607	21 und höher

Überprüfen Sie die Firmware- und FPGA-Versionen des TJ1-MC__.

Verbinden Sie den TJ1-MC__ mit der Trajexia Tools-Software. Weitere Informationen finden Sie im Programmierhandbuch.
Öffnen Sie das Terminalfenster, und geben Sie die folgenden Befehle ein:

Geben Sie `PRINT VERSION` in das Terminalfenster ein. Der Versionsparameter zeigt die aktuelle Firmware-Versionsnummer des Motion-Controllers an.
Geben Sie `PRINT FPGA_VERSION SLOT(-1)` in das Terminalfenster ein.
Der Parameter zeigt die aktuelle FPGA-Versionsnummer des TJ1-MC__ an.

1 Sicherheitswarnungen und -hinweise 16
1.1 Zielgruppe 16
1.2 Allgemeine Sicherheitshinweise 16
1.3 Sicherheits- hinweise 16
1.4 Sicherheitshinweise für die Betriebsumgebung 17
1.5 Sicherheitshinweise zu Anwendungen 18
1.6 Vorsichtsmaßnahmen bei der Montage von Baugruppen 21

2 Trajexia-System 22
2.1 Einführung 22
2.1.1 Trajexia Hardware 23
2.1.2 Dieses Handbuch 23
2.2 Multitasking-BASIC-Programmierung 23
2.3 BASIC-Programmierung 24
2.3.1 Achse, System und Task-Anweisungen 24
2.3.2 Speicherbereiche 24
2.3.3 Datenstrukturen und Variablen 25
2.3.4 Mathematische Spezifikationen 27
2.4 Bewegungsausführung 28
2.4.1 Profilgenerator 28
2.4.2 Sequenzsteuerung 29
2.4.3 Laden eines Bewegungsbefehls 29
2.5 Befehlszeilen-Schnittstelle 30
2.6 BASIC-Programme 30
2.6.1 Verwalten von Programmen 30
2.6.2 Programm-Kompilierung 31
2.6.3 Programmausführung 31

3 BASIC-Befehle 33
3.1 Kategorien 33
3.1.1 Achsenbefehle 33
3.1.2 Achsenparameter 34
3.1.3 Kommunikationsbefehle und -parameter 36
3.1.4 Konstanten 36
3.1.5 E/A-Befehle, -funktionen und -parameter 36
3.1.6 Mathematische Funktionen und Operanden 37
3.1.7 Programmbefehle 37
3.1.8 Programmsteuerungsbefehle 38
3.1.9 Slot-Parameter und -modifikatoren 38
3.1.10 Systembefehle und -funktionen 38

3.1.11 Systemparameter 39
3.1.12 Task-Befehle und -parameter 40
3.2 Alle BASIC-Befehle....41
3.2.1 + (Addition)....41
3.2.2 – (Subtraktion)....41
3.2.3 * (Multiplikation)....41
3.2.4 / (Division)....41
3.2.5 ^ (Potenz)....42
3.2.6 = (Ist gleich)....42
3.2.7 = (Zuordnung)....42
3.2.8 <> (Ist ungleich)....42
3.2.9 > (Ist größer als)....43
3.2.10 >= (Ist größer oder gleich)....43
3.2.11 < (Ist kleiner als)....43
3.2.12 <= (Ist kleiner oder gleich)....43
3.2.13 $ (Hexadezimale Eingabe)....44
3.2.14 ' (Kommentarfeld)....44
3.2.15 : (Anweisungs-Trennzeichen)....44
3.2.16 #....44
3.2.17 ABS....45
3.2.18 ACC....45
3.2.19 ACCEL....45
3.2.20 ACOS....45
3.2.21 ADD_DAC....46
3.2.22 ADDAX....46
3.2.23 ADDAX_AXIS....47
3.2.24 AIN....47
3.2.25 AND....47
3.2.26 AOUT....48
3.2.27 ASIN....48
3.2.28 ATAN....48
3.2.29 ATAN2....48
3.2.30 ATYPE....49
3.2.31 AUTORUN....49
3.2.32 AXIS....49
3.2.33 AXIS_DISPLAY....50
3.2.34 AXIS_ENABLE....50
3.2.35 AXISSTATUS....50
3.2.36 B_SPLINE....51
3.2.37 BASE....51

3.2.38 BASICERROR 52
3.2.39 BATTERY_LOW 53
3.2.40 BREAK_RESET 53
3.2.41 CAM 53
3.2.42 CAMBOX 55
3.2.43 CANCEL 56
3.2.44 CHECKSUM 56
3.2.45 CHR 56
3.2.46 CLEAR 57
3.2.47 CLEAR_BIT 57
3.2.48 CLEAR_PARAMS 57
3.2.49 CLOSE_WIN 57
3.2.50 CLUTCH_RATE 57
3.2.51 COMMSERROR 58
3.2.52 COMMSTYPE 58
3.2.53 COMPILE 58
3.2.54 CONNECT 59
3.2.55 CONSTANT 59
3.2.56 CONTROL 59
3.2.57 COPY 60
3.2.58 COS 60
3.2.59 CREEP 60
3.2.60 D_GAIN 60
3.2.61 D_ZONE_MAX 61
3.2.62 D_ZONE_MIN 61
3.2.63 DAC 61
3.2.64 DAC_OUT 61
3.2.65 DAC_SCALE 61
3.2.66 DATE 62
3.2.67 DATE$ 62
3.2.68 DATUM 62
3.2.69 DATUM_IN 63
3.2.70 DAY 64
3.2.71 DAY$ 64
3.2.72 DECEL 64
3.2.73 DEFPOS 64
3.2.74 DEL 65
3.2.75 DEMAND_EDGES 65
3.2.76 DEVICENET 65
3.2.77 DIR 67

3.2.78 DISABLE_GROUP 67
3.2.79 DISPLAY 67
3.2.80 DPOS 68
3.2.81 DRIVE_ALARM 68
3.2.82 DRIVE_CLEAR 69
3.2.83 DRIVE_CONTROL 69
3.2.84 DRIVE_INPUTS 70
3.2.85 DRIVE_MONITOR 70
3.2.86 DRIVE_READ 71
3.2.87 DRIVE_RESET 71
3.2.88 DRIVE_STATUS 72
3.2.89 DRIVE_WRITE 73
3.2.90 EDIT 73
3.2.91 ELSE 73
3.2.92 ELSEIF 73
3.2.93 ENCODER 74
3.2.94 ENCODER_BITS 74
3.2.95 ENCODER_CONTROL 74
3.2.96 ENCODER_ID 75
3.2.97 ENCODER_RATIO 75
3.2.98 ENCODER_READ 75
3.2.99 ENCODER_STATUS 76
3.2.100 ENCODER_TURNS 76
3.2.101 ENCODER_WRITE 76
3.2.102 ENDIF 76
3.2.103 ENDMOVE 77
3.2.104 EPROM 77
3.2.105 ERROR_AXIS 77
3.2.106 ERROR_LINE 77
3.2.107 ERRORMASK 78
3.2.108 ETHERNET 78
3.2.109 EX 79
3.2.110 EXP 79
3.2.111 FALSE 79
3.2.112 FAST_JOG 79
3.2.113 FASTDEC 80
3.2.114 FE 80
3.2.115 FE_LATCH 80
3.2.116 FE_LIMIT 80
3.2.117 FE_LIMIT_MODE 81

3.2.118 FE_RANGE 81
3.2.119 FHOLD_IN 81
3.2.120 FHSPEED 82
3.2.121 FINS_COMMS 82
3.2.122 FLAG 84
3.2.123 FLAGS 84
3.2.124 FOR..TO..STEP..NEXT 85
3.2.125 FORWARD 86
3.2.126 FPGA_VERSION 86
3.2.127 FRAC 86
3.2.128 FRAME 86
3.2.129 FREE 87
3.2.130 FS_LIMIT 87
3.2.131 FWD_IN 87
3.2.132 FWD_JOG 88
3.2.133 GET 88
3.2.134 GLOBAL 89
3.2.135 GOSUB..RETURN 89
3.2.136 GOTO 89
3.2.137 HALT 90
3.2.138 HEX 90
3.2.139 HLM_COMMAND 90
3.2.140 HLM_READ 91
3.2.141 HLM_STATUS 92
3.2.142 HLM_TIMEOUT 93
3.2.143 HLM_WRITE 94
3.2.144 HLS_NODE 95
3.2.145 HW_PSWITCH 95
3.2.146 I_GAIN 96
3.2.147 IDLE 96
3.2.148 IEEE_IN 96
3.2.149 IEEE_OUT 96
3.2.150 IF..THEN..ELSE..ENDIF 97
3.2.151 IN 97
3.2.152 INDEVICE 98
3.2.153 INITIALISE 98
3.2.154 INPUT 99
3.2.155 INT 99
3.2.156 INVERT_IN 99
3.2.157 INVERT_STEP 100

3.2.158 INVERTER_COMMAND 100
3.2.159 INVERTER_READ 101
3.2.160 INVERTER_WRITE 102
3.2.161 JOGSPEED 103
3.2.162 KEY 103
3.2.163 LAST_AXIS 104
3.2.164 LINKAX 104
3.2.165 LINPUT 104
3.2.166 LIST 105
3.2.167 LIST_GLOBAL 105
3.2.168 LN 106
3.2.169 LOCK 106
3.2.170 MARK 106
3.2.171 MARKB 107
3.2.172 MECHATROLINK 107
3.2.173 MERGE 108
3.2.174 MHELICAL 109
3.2.175 MOD 109
3.2.176 MOTION_ERROR 109
3.2.177 MOVE 110
3.2.178 MOVEABS 111
3.2.179 MOVECIRC 112
3.2.180 MOVELINK 114
3.2.181 MOVEMODIFY 115
3.2.182 MPOS 116
3.2.183 MSPEED 116
3.2.184 MTYPE 116
3.2.185 NAIO 117
3.2.186 NEG_OFFSET 117
3.2.187 NEW 117
3.2.188 NEXT 117
3.2.189 NIO 117
3.2.190 NOT 118
3.2.191 NTYPE 118
3.2.192 OFF 118
3.2.193 OFFPOS 118
3.2.194 ON 119
3.2.195 ON.. GOSUB 119
3.2.196 ON.. GOTO 119
3.2.197 OP 119

3.2.198 OPEN_WIN 120
3.2.199 OR 120
3.2.200 OUTDEVICE 121
3.2.201 OUTLIMIT 121
3.2.202 OV_GAIN 121
3.2.203 P_GAIN 122
3.2.204 PI 122
3.2.205 PMOVE 122
3.2.206 POS_OFFSET 123
3.2.207 POWER_UP 123
3.2.208 PRINT 123
3.2.209 PROC 124
3.2.210 PROC_STATUS 124
3.2.211 PROCESS 125
3.2.212 PROCNUMBER 125
3.2.213 PROFIBUS 125
3.2.214 PSWITCH 126
3.2.215 RAPIDSTOP 127
3.2.216 READ_BIT 127
3.2.217 REG_POS 128
3.2.218 REG_POSB 128
3.2.219 REGIST 128
3.2.220 REMAIN 130
3.2.221 REMOTE_ERROR 130
3.2.222 RENAME 130
3.2.223 REP_DIST 131
3.2.224 REP_OPTION 131
3.2.225 REPEAT..UNTIL 132
3.2.226 RESET 132
3.2.227 RETURN 132
3.2.228 REV_IN 132
3.2.229 REV_JOG 133
3.2.230 REVERSE 133
3.2.231 RS_LIMIT 133
3.2.232 RUN 133
3.2.233 RUN_ERROR 134
3.2.234 RUNTYPE 134
3.2.235 S_REF 135
3.2.236 S_REF_OUT 135
3.2.237 SCOPE 136

3.2.238 SCOPE_POS 137
3.2.239 SELECT 137
3.2.240 SERVO 137
3.2.241 SERVO_PERIOD 137
3.2.242 SET_BIT 138
3.2.243 SETCOM 138
3.2.244 SGN 139
3.2.245 SIN 139
3.2.246 SLOT 139
3.2.247 SPEED 139
3.2.248 SQR 140
3.2.249 SRAMP 140
3.2.250 STEP 140
3.2.251 STEP_RATIO 140
3.2.252 STEPLINE 141
3.2.253 STOP 141
3.2.254 SYSTEM_ERROR 142
3.2.255 T_REF 142
3.2.256 TABLE 143
3.2.257 TABLEVALUES 143
3.2.258 TAN 144
3.2.259 THEN 144
3.2.260 TICKS 144
3.2.261 TIME 144
3.2.262 TIME$ 145
3.2.263 TO 145
3.2.264 TRANS_DPOS 145
3.2.265 TRIGGER 145
3.2.266 TROFF 145
3.2.267 TRON 146
3.2.268 TRUE 146
3.2.269 TSIZE 146
3.2.270 UNITS 147
3.2.271 UNLOCK 147
3.2.272 UNTIL 147
3.2.273 VERIFY 147
3.2.274 VERSION 147
3.2.275 VFF_GAIN 148
3.2.276 VP_SPEED 148
3.2.277 VR 148

3.2.278 VRSTRING 149
3.2.279 WA 149
3.2.280 WAIT IDLE 150
3.2.281 WAIT LOADED 150
3.2.282 WAIT UNTIL 150
3.2.283 WDOG 151
3.2.284 WHILE..WEND 151
3.2.285 XOR 152

4 Kommunikationsprotokolle 153
4.1 Verfügbare Schnittstellen 153
4.2 Ethernet 153
4.2.1 Kommunikation mit Trajexia direkt vom PC aus 154
4.2.2 Dezentrale Kommunikation mit Trajexia 155
4.2.3 Trajexia Tools-Protokoll 156
4.2.4 FINS-Serverprotokoll 156
4.2.5 FINS-Client-Protokoll 158
4.3 Serielles Protokoll 158
4.3.1 Host-Link-Master 158
4.3.2 Host-Link-Slave 163
4.3.3 Benutzerdefiniertes Protokoll 165
4.4 PROFIBUS 167
4.4.1 Einführung 167
4.4.2 Einrichtung der Kommunikation 167
4.4.3 Kommunikationsstatus 172
4.5 DeviceNet 173
4.5.1 Einführung 173
4.5.2 Einrichtung der Kommunikation 173
4.5.3 Kommunikationsstatus 178
4.6 MECHATROLINKII 179

5 Trajexia Tools-Schnittstelle 180
5.1 Einführung 180
5.2 Spezifikationen und Anschlüsse 180
5.2.1 PC-Spezifikationen 180
5.2.2 Installieren der Trajexia-Tool-Software 181
5.2.3 Anschließen an den TJ1-MC__ 186
5.3 Projekte 191
5.3.1 Trajexia Tools Projects 191
5.3.2 Das Fenster „Projekt überprüfen“ 192
5.4 Trajexia-Tools-Anwendungsfenster 194

5.4.1 Bedienfeld....194
5.4.2 Menüleiste....194
5.4.3 Werkzeugleiste....195
5.5 Menübeschreibungen....196
5.5.1 Projektmenü....196
5.5.2 Steuerungsmenü....198
5.5.3 Programm-Menü....203
5.5.4 Tools-Menü....205
5.5.5 Das Menü „Options“ (Optionen)....221
5.5.6 Das Menü „Windows“ (Anzeige)....224
5.5.7 Das Menü „Help“ (Hilfe)....224

6 Beispiele und Tipps....225
6.1 Anleitungen....225
6.1.1 STARTUP-Programm....225
6.1.2 Verstärkungseinstellungen....229
6.1.3 Einstellung des Achsenparameters UNITS und des Übersetzungsverhältnisses....239
6.1.4 Zuordnung der Servotreiber-Eingänge und -Ausgänge....251
6.1.5 Nullpunktsuche....253
6.1.6 Registrierung....259
6.1.7 Datenaufzeichnung und Überwachung....269
6.2 Praktische Beispiele....279
6.2.1 Shell-Programm....279
6.2.2 Initialisierungsprogramm....283
6.2.3 Programm für eine Achse....286
6.2.4 Position mit Produkterkennung....287
6.2.5 Position in einem Gitter....289
6.2.6 Tüten-Zuführprogramm....291
6.2.7 CAM-Tabelle innerhalb eines Programms....294
6.2.8 Schneidvorrichtungsprogramm....295
6.2.9 Korrekturprogramm....298

7 Fehlersuche....300
7.1 Spannung und Analyse-Tools....300
7.2 TJ1-MC__....300
7.2.1 Systemfehler....300
7.2.2 Achsenfehler....300
7.2.3 Baugruppenfehler....301
7.2.4 Konfigurationsfehler....302
7.2.5 Austauschen der Batterie....302
7.3 TJ1-PRT....302

7.3.1 Systemfehler....302
7.3.2 E/A-Datenübertragungsprobleme....303
7.4 TJ1-DRT....304
7.4.1 Systemfehler....304
7.4.2 E/A-Datenübertragungsprobleme....304
7.5 TJ1-ML__....304
7.5.1 Systemfehler....304
7.5.2 Bus-Fehler....304
7.6 TJ1-FL02....305
7.6.1 Systemfehler....305

1 Sicherheitswarnungen und -hinweise

1.1 Zielgruppe

Dieses Handbuch ist an Personen gerichtet, die Kenntnisse von elektrischen Systemen (z.B. Elektrotechniker o. ä.) haben und die für die Entwicklung, Installation und Verwaltung von Fabrikautomationssystemen und -einrichtungen verantwortlich sind.

1.2 Allgemeine Sicherheitshinweise

Der Anwender darf das Produkt nur entsprechend den in diesem Handbuch niedergelegten Vorgaben einsetzen.
Wenden Sie sich vor der Verwendung dieses Produktes an Ihren OMRON-Vertreter, falls Sie das Produkt unter Bedingungen verwenden, die nicht im Bedienerhandbuch aufgeführt sind bzw. wenn Sie das Produkt im Bereich der Nukleartechnik, im Eisenbahnverkehr, in der Luftfahrt, in Fahrzeugen, in Sicherheitseinrichtungen, petrochemischen Fabriken oder anderen Systemen, Geräten oder Ausrüstungen verwenden möchten, bei denen fehlerhafte Verwendung zu schwerwiegenden Gefahren für Leben und Sachgut führen kann.

1.3 Sicherheits- hinweise

VORSICHT
Zerlegen Sie das Gerät nicht und berühren Sie keine inneren Teile, solange die Spannungsversorgung eingeschaltet ist.
Es besteht die Gefahr eines elektrischen Schlages.

VORSICHT
Berühren Sie Klemmen oder Klemmenblöcke nicht bei eingeschalteter Versorgungsspannung.
Es besteht die Gefahr eines elektrischen Schlags.

VORSICHT
Die Anschlüsse der Plus- und Minuspole der Batterien dürfen nicht kurzgeschlossen werden, und die Batterien dürfen nicht aufgeladen, zerlegt, durch Druck deformiert oder verbrannt werden.
Die Batterien können explodieren, brennen, oder es kann Flüssigkeit austreten.

VORSICHT
Der Kunde muss durch entsprechende Vorkehrungen die Systemsicherheit für den Fall sicherstellen, dass bedingt durch unterbrochene Signalleitungen, vorübergehende Spannungsausfälle oder beliebige andere Ursachen falsche, fehlende oder ungewöhnliche Signale auftreten.
Andernfalls besteht die Gefahr von schweren Unfällen.

VORSICHT
Der Anwender muss Not-Aus-Schaltungen, Verriegelungs- und Begrenzungsschaltungen sowie ähnliche Sicherheitsmaßnahmen in externen Schaltungen (d. h. nicht im Trajexia Motion-Controller) vorsehen.
Andernfalls besteht die Gefahr von schweren Unfällen.

VORSICHT
Wenn der 24 V DC-Ausgang (E/A-Spannungsversorgung für den TJ1) überlastet oder kurzgeschlossen ist, kann die Spannung abfallen, was eine Abschaltung der Ausgänge zur Folge hat. Um solche Probleme zu vermeiden, müssen externe Sicherheitsvorrichtungen installiert werden, um die Sicherheit des Systems zu gewährleisten.

VORSICHT
Die TJ1-Ausgänge werden abgeschaltet, wenn die Ausgangstransistoren überlastet sind (Schutzvorrichtung). Um solche Probleme zu vermeiden, müssen externe Sicherheitsvorrichtungen installiert werden, um die Sicherheit des Systems zu gewährleisten.

VORSICHT
Der TJ1 schaltet das WDOG ab, wenn seine Selbstdiagnosefunktion einen Fehler erkennt. Um solche Fehler zu vermeiden, müssen externe Sicherheitsvorrichtungen installiert werden, um die Sicherheit des Systems zu gewährleisten.

VORSICHT
Statten Sie externe Schaltkreise (d. h. nicht die im Trajexia Motion-Controller, hier als „TJ1“ bezeichnet) mit Sicherheitsvorrichtungen aus, damit die Sicherheit im System gewährleistet bleibt, wenn aufgrund einer Fehlfunktion des TJ1 oder eines anderen externen Faktors mit Auswirkungen auf den TJ1-Betrieb eine Störung auftritt. Andernfalls besteht die Gefahr von schweren Unfällen.

VORSICHT
Versuchen Sie nicht, Baugruppen zu zerlegen, instandzusetzen oder zu modifizieren.
Bei jedem Versuch besteht das Risiko von Fehlfunktionen sowie die Gefahr eines elektrischen Schlags bzw. Brandgefahr.

Achtung
Überprüfen Sie die Sicherheit in der Zielbaugruppe, bevor Sie ein Programm in eine andere Baugruppe übertragen oder den Speicher bearbeiten.
Wenn Sie eine dieser Aktionen durchführen, ohne die Sicherheit zu überprüfen, kann dies zu Verletzungen führen.

Achtung
Anwenderprogramme, die in die Motion Controller-Baugruppe geschrieben werden, werden nicht automatisch im Flash-Speicher (Flash-Speicherfunktion) des TJ1 gesichert.

Achtung
Beachten Sie beim Verkabeln der DC-Spannungsversorgung auf die Polarität (+/–). Eine falsche Polung kann Fehlfunktionen des Systems verursachen.

Achtung
Ziehen Sie die Schrauben am Klemmenblock der Spannungsversorgungsbaugruppe mit dem in diesem Handbuch angegebenen Drehmoment fest.
Lose Schrauben können zu Schmorbränden oder Fehlfunktionen führen.

1.4 Sicherheitshinweise für die Betriebsumgebung

Achtung
Betreiben Sie die Baugruppe nicht an den folgenden Orten: Andernfalls besteht die Gefahr von Fehlfunktionen, elektrischen Schlägen oder Brandgefahr.

- Orte, die direkter Sonneneinstrahlung ausgesetzt sind.
- Orte, an denen Temperaturen oder Luftfeuchtigkeit außerhalb der in den technischen Daten angegebenen Bereiche herrschen.
- Orte, die starken Temperaturschwankungen und damit Kondensatbildung ausgesetzt sind.
- Orte, an denen das Gerät korrosiven oder entzündlichen Gasen ausgesetzt ist.
- Orte, die dem Einfluss von Stäuben (besonders Eisenstaub) oder Salzen ausgesetzt sind.
- Orte, die dem Einfluss von Feuchtigkeit, Öl oder Chemikalien ausgesetzt sind.
- Orte, die Stößen oder Schwingungen ausgesetzt sind.

Achtung
Ergreifen Sie beim Installieren von Systemen an folgenden Orten angemessene und geeignete Gegenmaßnahmen. Ungeeignete und unzureichende Maßnahmen können zu Fehlfunktionen führen.

- Orte mit statischer Aufladung und anderen Störungen.
- Orte mit starken elektromagnetischen Feldern.
- Orte, die möglicherweise Radioaktivität ausgesetzt sind.
- Orte in der Nähe von Spannungsversorgungen.

Achtung
Die Betriebsumgebung des TJ1-Systems kann große Auswirkungen auf die Lebensdauer und Zuverlässigkeit des Systems haben. Ungeeignete Betriebsumgebungen können Fehlfunktionen, Ausfälle und andere unvorhergesehene Probleme in Bezug auf das TJ1-System zur Folge haben.
Achten Sie darauf, dass die Betriebsumgebung zum Zeitpunkt der Installation die angegebenen Bedingungen erfüllt und dass diese Bedingungen während der Lebensdauer des Systems erfüllt bleiben.

1.5 Sicherheitshinweise zu Anwendungen

VORSICHT
Prüfen Sie zuerst, ob die Achsen vorhanden und vom richtigen Typ sind, bevor Sie das System starten.
Die Anzahl der flexiblen Achsen ändert sich, wenn beim Aufstarten des Controllers MECHATROLINK-II-Netzwerkfehler auftreten oder wenn sich die MECHATROLINK-II-Netzwerkkonfiguration ändert.

VORSICHT
Überprüfen Sie das Anwenderprogramm auf ordnungsgemäße Ausführung, bevor Sie es tatsächlich für die Baugruppe einsetzen. Wird das Programm nicht überprüft, besteht die Gefahr von unerwartetem Verhalten.

Achtung
Verwenden Sie stets die diesem Handbuch angegebene Versorgungsspannung.
Eine falsche Spannung kann zu Fehlfunktionen oder Brandgefahr führen.

Achtung
Ergreifen Sie geeignete Maßnahmen, um sicherzustellen, dass das Gerät mit der angegebenen Nennspannung und -frequenz versorgt wird. Lassen Sie an Installationsorten, an denen die Spannungsversorgung nicht stabil ist, besondere Vorsicht walten. Eine falsche Spannungsversorgung kann zu Fehlfunktionen führen.

Achtung
Installieren Sie externe Trennschalter, und ergreifen Sie weitere Sicherheitsmaßnahmen gegen Kurzschlüsse in externer Verdrahtung. Bei unzureichenden Sicherheitsmaßnahmen gegen Kurzschlüsse besteht Brandgefahr.

Achtung
Legen Sie keine Spannung an die Eingangsbaugruppe an, die die Nenneingangsspannung übersteigt.
Bei zu hoher Spannung besteht Brandgefahr.

Achtung
Legen Sie keine Spannung an die Ausgangsbaugruppen an und schließen Sie keine Lasten an die Ausgangsbaugruppen an, die die maximale Schaltleistung übersteigen.
Bei übermäßigen Spannungen oder Lasten besteht Brandgefahr.

Achtung
Klemmen Sie die Erdung ab, wenn Sie Spannungsfestigkeitstests durchführen.
Wird die Erdung nicht abgeklemmt, besteht Brandgefahr.

Achtung
Achten Sie bei der Installation der Baugruppen auf eine Erdung der Klasse 3 (bis max. 100Ω).
Wenn Sie keine Erdung der Klasse 3 anschließen, besteht die Gefahr eines elektrischen Schlags.

Achtung
Schalten Sie die Spannungsversorgung für das System immer AUS, bevor Sie versuchen, eine der folgenden Arbeiten durchzuführen.
Bei Nichtabschalten der Versorgungsspannung besteht das Risiko von Fehlfunktionen und die Gefahr eines elektrischen Schlags.

- Anbringen oder Abbauen von Erweiterungsbaugruppen, CPU-Baugruppen oder sonstigen Baugruppen.
- Montieren der Baugruppen.
- Einstellen der DIP-Schalter oder Drehschalter.
- Anschließen oder Verdrahten von Kabeln.
- Anschließen oder Lösen von Steckverbindungen.

Achtung
Stellen Sie sicher, dass alle Befestigungsschrauben, Klemmenschrauben und Schrauben von Steckverbindern mit dem in diesem Handbuch angegebenen Drehmoment festgezogen sind.
Ein falsches Anzugsdrehmoment kann zu Fehlfunktionen führen.

Achtung
Belassen Sie beim Verdrahten die Schutzfolie an der Baugruppe.
Bei entfernter Schutzfolie besteht die Gefahr von Fehlfunktionen.

Achtung
Entfernen Sie die Schutzfolie nach Abschluss der Verdrahtungsarbeiten, um eine ordnungsgemäße Wärmeableitung zu gewährleisten.
Bei nicht entfernter Schutzfolie besteht die Gefahr von Fehlfunktionen.

Achtung
Verwenden Sie für die Verkabelung Crimp-Kabelschuhe.
Schließen Sie keine abisolierten Litzendrähte direkt an Klemmen an. Beim Anschließen frei liegender Litzen besteht Brandgefahr.

Achtung
Überprüfen Sie alle Verdrahtungen sorgfältig, bevor Sie die Spannungsversorgung einschalten.
Bei unsachgemäßer Verdrahtung besteht Brandgefahr.

Achtung
Achten Sie auf die richtige Verdrahtung.
Bei unsachgemäßer Verdrahtung besteht Brandgefahr.

Achtung
Installieren Sie Baugruppe erst, nachdem Sie den Klemmenblock eingehend überprüft haben.

Achtung
Stellen Sie sicher, dass die Klemmenblöcke, Verlängerungskabel und andere Teile mit Arretierungen ordnungsgemäß eingerastet sind.
Bei fehlender Arretierung besteht die Gefahr von Fehlfunktionen.

Achtung
Vergewissern Sie sich, dass keine negativen Auswirkungen im System auftreten können, bevor Sie die Betriebsart des Systems ändern.
Andernfalls besteht die Gefahr von unerwartetem Verhalten.

Achtung
Nehmen Sie den Betrieb erst dann wieder auf, wenn der für den Betrieb benötigte VR- und Tabellenspeicher auf die neue CPU-Baugruppe übertragen wurde.
Andernfalls besteht die Gefahr von unerwartetem Verhalten.

Achtung
Achten Sie beim Austauschen von Komponenten darauf, dass die neue Komponente die richtigen Nenndaten besitzt.
Andernfalls besteht die Gefahr von Fehlfunktionen oder Brandgefahr.

Achtung
Ziehen Sie nicht an den Kabeln, und biegen Sie die Kabel nicht über das natürliche Maß hinaus. Die Kabel können dabei brechen.

Achtung
Achten Sie vor Berühren des Systems darauf, dass Sie zunächst einen geerdeten Metallgegenstand berühren, um statische Aufladung abzuleiten.
Andernfalls besteht die Gefahr von Fehlfunktionen oder Schäden.

Achtung
Die UTP-Kabel sind nicht abgeschirmt. Verwenden Sie in Umgebungen, in denen elektrische Störungen auftreten, ein System mit abgeschirmten, paarweise verdrillten Kabeln (STP-Kabeln) und Hubs, die für eine FA-Umgebung geeignet sind.
Schließen Sie keine verdrillten Kabel an Hochspannungsleitungen an.
Schließen Sie keine verdrillten Kabel in der Nähe von Geräten an, die elektrische Störungen erzeugen.
Schließen Sie keine verdrillten Kabel an Orten mit hoher Luftfeuchtigkeit an.
Verwenden Sie keine verdrillten Kabel an Orten, an denen übermäßig viel Schmutz oder Staub, Ölnebel oder andere Schmutzstoffe auftreten.

Achtung
Verwenden Sie die in den Betriebshandbüchern angegebenen Spezialkabel zum Anschluss der Baugruppen. Die Verwendung von handelsüblichen RS-232C-Computerkabeln kann zu Fehlfunktionen in externen Geräten oder der Motion Controller-Baugruppe führen.

Achtung
Aufgrund einer Fehlfunktion in den eingebauten Transistorausgängen oder anderen internen Schaltkreisen werden die Ausgänge möglicherweise nicht abgeschaltet. Um solche Probleme zu vermeiden, müssen externe Sicherheitsvorrichtungen installiert werden, und die Sicherheit des Systems zu gewährleisten.

Achtung
Der TJ1 startet in der RUN-Betriebsart, wenn der Strom eingeschaltet wird und wenn ein BASIC-Programm auf die Betriebsart „Auto Run“ eingestellt ist.

1.6 Vorsichtsmaßnahmen bei der Montage von Baugruppen

Achtung
Installieren Sie die Baugruppe ordnungsgemäß.
Bei unsachgemäßer Installation der Baugruppe besteht die Gefahr von Fehlfunktionen.

Achtung
Montieren Sie die mit dem TJ1-MC__ gelieferte Abschlussbaugruppe an der rechten äußeren Baugruppe.
Der TJ1 kann erst ordnungsgemäß funktionieren, wenn die Abschlussbaugruppe richtig montiert ist.

2 Trajexia-System

2.1 Einführung

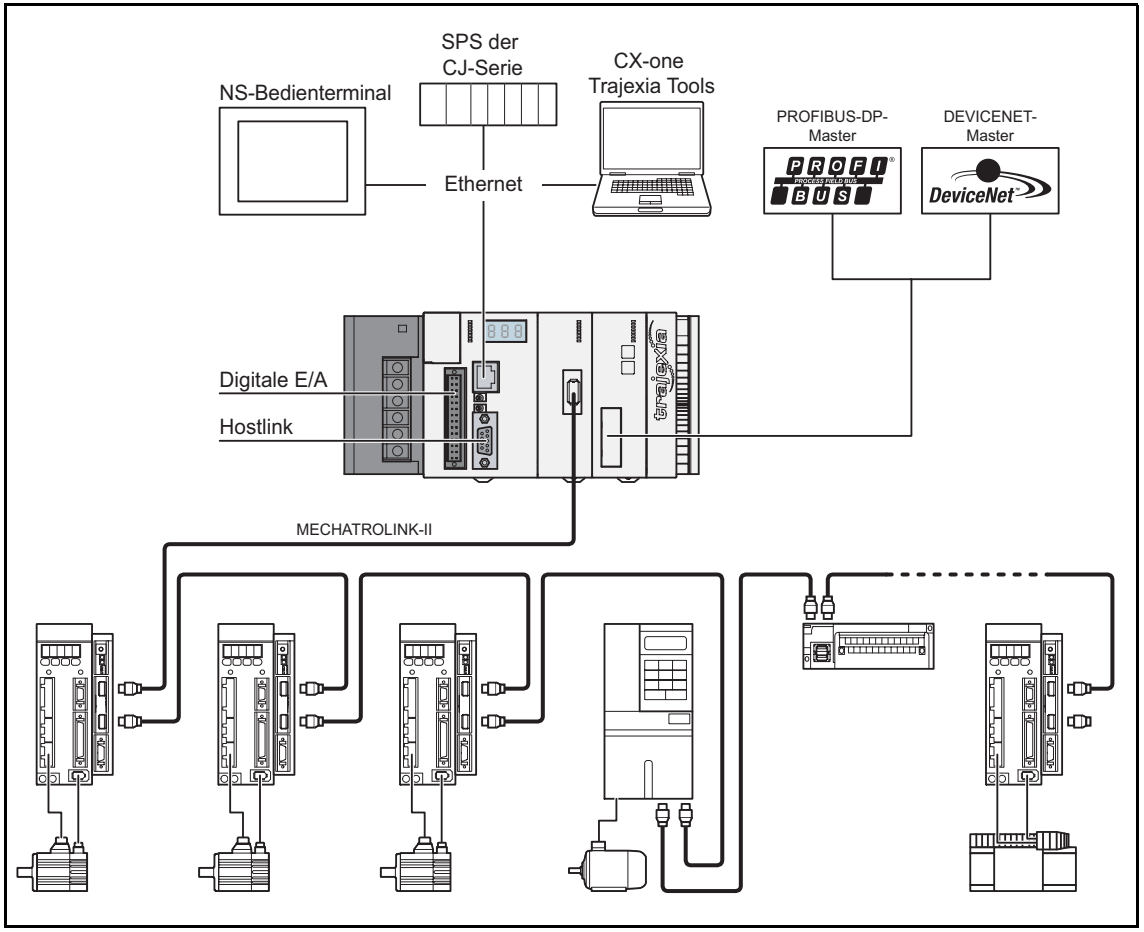
Trajexia ist OMRON's Plattform zur Bewegungssteuerung, die Ihnen die Leistung und leichte Bedienbarkeit eines modernen Motion Control-Systems bietet.

Trajexia ist ein modulares Standalone-System, das maximale Flexibilität und Skalierbarkeit erlaubt. Herzstück des Trajexia ist die TJ1-Multi-Tasking-Motion-CPU. Durch einen 32-Bit-DSP ist sie speziell zur Bewältigung von Bewegungsaufgaben mit einfachen Motion-Befehlen prädestiniert, z. B. E-Cam, E-Gearbox, Druckmarkensteuerung und Interpolation.

Trajexia ermöglicht die Steuerung von bis zu 16 Achsen über einen MECHATROLINK-II Motion-Bus oder über herkömmliche Analog- oder Impulssteuerung mit unabhängiger Positionierung, Drehzahl- oder Drehmomentregelung für jede Achse.
Der leistungsfähige Motion-Befehlssatz ermöglicht zudem eine intuitive und einfache Programmierung.

Eine große Auswahl an erstklassigen Servo-, Linear- und Torquemotoren sowie Servotreibern und Frequenzumrichtern steht für die Lösung Ihrer Applikation zur Verfügung.
Das System kann bis auf sechzehn Achsen und acht Frequenzumrichter und E/A-Module skaliert werden.

Abb.1



2.1.1 Trajexia Hardware

Die Trajexia Hardware wird im Trajexia Hardware-Referenzhandbuch näher erläutert. Es ist empfehlenswert, das Hardware-Referenzhandbuch vorab zu lesen. Das Trajexia-System bietet folgende Vorteile:

Direkte Verbindung über Ethernet
Die integrierte Ethernet-Schnittstelle des Trajexia ermöglicht die direkte und schnelle Verbindung mit PC's, SPS, MMI's und anderen Geräten. Der direkte Zugriff auf die Antriebsparameter ist über den schnellen Motion-Bus MECHATROLINK-II gewährleistet. Trajexia ermöglicht „Explicit Messaging" über Ethernet und den MECHATROLINK-II Motion-Bus. Dies bietet eine vollständige Transparenz bis hinunter zur Aktuatorenebene und ermöglicht einen dezentralen Zugriff.

Schutz Ihres Know-Hows
Ein Passwortschutz nach einem Verschlüsselungsverfahren garantiert den zuverlässigen Schutz Ihres wertvollen Know-Hows.

Serielle Schnittstelle und lokale E/A's
Eine serielle Schnittstelle ermöglicht den direkten Anschluss an jede beliebige OMRON-SPS- und MMI-Systeme sowie an andere Feldgeräte. Dank der frei konfigurierbaren integrierten 16 Eingänge und 8 Ausgänge des Controllers können Sie den Trajexia perfekt in Ihr Anlagenkonzept integrieren.

MECHATROLINK-II-Master
Der MECHATROLINK-II-Master steuert bis zu 16 Servos, Frequenzumrichter oder E/A und ermöglicht eine vollständige Transparenz des gesamten Systems. MECHATROLINK-II bietet die Kommunikationsgeschwindigkeit und Zeitgenauigkeit, die für eine perfekte Bewegungssteuerung von Servos erforderlich sind. Die Bewegungszykluszeit kann zwischen 0,5 ms, 1 ms oder 2 ms gewählt werden.

TJ1-FL02 (Flex Achsmodul-Baugruppe)
Die TJ1-FL02 bedient alle analogen Antriebe und Schrittmotorsteuerungen. Das Modul unterstützt die wesentlichen Encoder-Protokolle und erlaubt so den Anschluss eines externen Encoders an das System.

Antriebe und Frequenzumrichter
Eine große Auswahl an Servo-, Linear- und Torquemotoren sowie Servotreibern und Frequenzumrichtern, die Ihren Anforderungen an Integrationsdichte, Leistung und Zuverlässigkeit entsprechen, steht Ihnen für die Lösung Ihrer Applikation zur Verfügung. Die an den MECHATROLINK-II Motion-Bus angeschlossenen Frequenzumrichter werden mit derselben Systemtaktzeit wie die Servoantriebe betrieben.

Dezentrale E/A's
Dezentrale E/A-Module am MECHATROLINK-II Motion-Bus ermöglichen eine Erweiterung des Systems bei Beibehaltung eines einzigen durchgängigen Motion-Bussystems.

PROFIBUS DP
Der PROFIBUS-DP-Slave erlaubt den Anschluss an ein PROFIBUS-Netzwerk in Ihrer Maschine.

DeviceNet
Der DeviceNet-Slave erlaubt den Anschluss an ein DeviceNet-Netzwerk in Ihrer Maschine.

2.1.2 Dieses Handbuch

Dieses Programmierhandbuch enthält spezielle Informationen zu folgenden Themen:
- Beschreibung und Verwendung der BASIC-Befehle
- Die für das Trajexia-System erforderlichen Kommunikationsprotokolle
- Die Verwendung und Beschreibung der Elemente der Trajexia Tools-Schnittstelle
- Programmbeispiele und gute Programmiertechnik
- Fehlersuche und Fehlerlokalisierung

2.2 Multitasking-BASIC-Programmierung

Die TJ1-MC__-Baugruppen (Motion Controller-Baugruppe) verfügen über eine Multitasking-Version der Programmiersprache BASIC. Die Motion Control-Sprache basiert zum großen Teil auf einem in Tokens übersetzten BASIC, und die Programme werden vor ihrer Ausführung in die Token-Form kompiliert.

Multitasking ist einfach einzurichten und erlaubt die Programmierung sehr komplexer Maschinen. Multitasking gibt dem TJ1-MC__ einen erheblichen Vorteil gegenüber gleichwertigen Singletask-Systemen. Es ermöglicht modulare Anwendungen, bei denen die logisch miteinander verbundenen Prozesse in dem gleichen Task-Programm zusammengruppiert werden können, wodurch die Architektur und der Aufbau des Codes vereinfacht werden.
Der TJ1-MC__ kann je nach Speichergröße bis zu 14 Programme verwalten. Die Ausführung der Programme wird vom Anwender über BASIC gesteuert. Die hier vorgestellten BASIC-Befehle, -funktionen und -parameter sind in Kapitel 3 enthalten.

2.3 BASIC-Programmierung

Die BASIC-Sprache besteht unter anderem aus Befehlen, Funktionen und Parametern. Diese BASIC-Anweisungen sind die Bausteine zur Betriebssteuerung des TJ1-MC__.
Befehle sind vom Prozessor erkannte Wörter, die eine bestimmte Handlung bewirken, aber keinen Wert zurückgeben. Beispielsweise ist **PRINT** ein erkanntes Wort, das bewirkt, dass der Wert der folgenden Funktionen oder Variablen auf einem bestimmten Ausgabegerät ausgedruckt wird.
Funktionen sind vom Prozessor erkannte Wörter, die eine bestimmte Aktion bewirken und einen mit der Aktion verknüpften Wert zurückgeben. Beispielsweise wird mit **ABS** der Wert des Parameters erfasst und der absolute Wert hiervon zur Verwendung durch eine andere Funktion oder einen anderen Befehl zurückgegeben. Zum Beispiel gibt **ABS(–1)** den Wert 1 zurück, der dann z. B. vom **PRINT** -Befehl benutzt werden kann, um eine Zeichenkette zu erzeugen, die auf einem bestimmten Gerät ausgegeben wird.
Parameter sind vom Prozessor erkannte Wörter, die einen bestimmten Wert enthalten. Dieser Wert kann gelesen und ggf. geschrieben werden, wenn Schreibzugriff möglich ist. Parameter werden verwendet, um das Verhalten des Systems zu bestimmen und zu überwachen. Beispielsweise bestimmt **ACCEL** die Beschleunigungsrate einer Bewegung für eine bestimmte Achse.

2.3.1 Achse, System und Task-Anweisungen

Die Befehle, Funktionen und Parameter gelten entweder für die Achse (n), die laufenden Tasks oder das gesamte System.

Achsenanweisungen

Die Bewegungssteuerungsbefehle und die Achsenparameter gelten für eine oder mehrere Achsen. Achsenparameter bestimmen und überwachen, wie eine Achse auf gegebene Befehle und auf die äußere Umgebung reagiert. Jede Achse hat einen eigenen Parametersatz, sodass alle Achsen unabhängig voneinander eingestellt werden können. Mit den Bewegungssteuerungsbefehlen lassen sich eine oder mehrere der Achsen gleichzeitig steuern, wobei jede Achse ihr eigenes Verhalten hat. Bei jedem Hochfahren werden die Achsenparameter auf ihre Standardwerte zurückgesetzt.
Die Befehle und Parameter wirken sich auf eine Basisachse oder eine Achsengruppe aus, die durch den Befehl **BASE** spezifiziert wird. Der Befehl **BASE** wird verwendet, um die Basisachsengruppe zu ändern; jeder Task hat seine eigene Gruppe, die jederzeit geändert werden kann. Die Standard-Basisachse ist 0.
Einzelne achsenabhängige Befehle oder Parameter können auch so programmiert werden, dass sie sich temporär auf eine Achse auswirken, indem die Funktion **AXIS** als Modifikator in den achsenabhängigen Befehl aufgenommen wird. Eine temporäre Achsenbasis wirkt sich nur auf den Befehl oder Parameter aus, nach dem **AXIS** eingegeben wurde.

Task-Anweisungen

Die Task-Parameter gelten für eine einzelne Task. Die Task-Parameter überwachen den Task, z. B. bei der Fehlerbehandlung. Der Modifikator **PROC** erlaubt es dem Anwender, auf einen Parameter eines Tasks zuzugreifen. Ohne **PROC** wird der aktuelle Task ausgeführt. Der Befehl **BASE** (siehe oben) ist Task-spezifisch und kann zusammen mit dem Modifikator **PROC** verwendet werden.

Systemanweisungen

Diese Anweisungen regeln die gesamten Funktionen des Systems, wobei es sich im Wesentlichen um alle Anweisungen handelt, die nicht zu den ersten beiden Gruppen gehören.

2.3.2 Speicherbereiche

Die Trajexia Motion Controller-Baugruppe verfügt über drei verschiedene Speicherbereiche:
- E/A-Speicher
- VR-Speicher
- TABLE-Speicher

E/A-Speicher

Der E/A-Speicher dient zur Speicherung des Zustands von Eingangs- und Ausgangsgeräten, die an das Trajexia-System angeschlossen sind. Er ist in zwei Unterbereiche unterteilt: einen für den digitalen E/A-Speicher und einen für den analogen E/A-Speicher. Der digitale E/A-Speicher enthält die Eingangs- und Ausgangszustände digitaler E/A-Geräte. Seine Kapazität umfasst 256 Bit (Eingangspunkte) für Eingänge und 256 Bit (Ausgangspunkte) für Ausgänge. Der Zugriff auf die Eingänge in diesem Speicher erfolgt über den Befehl **IN**. Der Zugriff auf die Ausgänge erfolgt über den Befehl **OUT**. Der analoge E/A-Speicher enthält die Ein- und Ausgangswerte von analogen E/A-Geräten. Seine Kapazität umfasst 36 Eingangs- und 36 Ausgangskanäle. Der Zugriff auf die analogen Eingangskanäle erfolgt über den Befehl **AIN**. Der Zugriff auf die analogen Ausgangskanäle erfolgt über den Befehl **AOUT**.

VR-Speicher

Der VR-Speicher wird häufig dann verwendet, wenn Daten oder Werte global verfügbar sein sollen, d. h., sie sollen gleichzeitig von allen Programmen in dem Projekt abrufbar sein. Die Größe dieses Speichers beträgt 1024 Slots mit Indizes von 0 bis 1023. Ein Speicher-Slot wird mit dem Makro **VR(x)** angesprochen, wobei **x** für den Index des VR-Speicher-Slots steht. Der VR-Speicher erlaubt Lese- und Schreibzugriff. Das Schreiben erfolgt über eine mathematische Anweisung mit dem Befehl **=** im Programm. Der Inhalt dieses Speichers wird in dem batteriegespeisten RAM-Speicher gehalten und wird auch bei abgeschalteter Versorgungsspannung nicht gelöscht. Der VR-Speicher bleibt auch beim Batteriewechsel erhalten, wenn dies schnell genug erfolgt.

TABLE-Speicher

Der TABLE-Speicher wird häufig dann verwendet, wenn Daten oder Werte global verfügbar sein sollen, d. h., sie sollen gleichzeitig von allen Programmen in dem Projekt abrufbar sein. Obwohl der VR-Speicher zu ähnlichen Zwecken verwendet wird, um verschiedene globale Daten und Werte zu definieren, wird der TABLE-Speicher für viel größere Mengen globaler Daten benutzt, die außerdem in einer bestimmten Reihenfolge angeordnet sein müssen. Aus diesem Grund wird der TABLE-Speicher normalerweise zur Speicherung von Tabellendaten, Bewegungsprofilen, Anmeldedaten usw. benutzt. Einige BASIC-Befehle, die diese Art und Größe von Daten bereitstellen, z. B. **SCOPE**, **CAM**, **CAMBOX** usw., erfordern die Verwendung des TABLE-Speichers zum Schreiben von Ergebnissen. Die Größe dieses Speichers beträgt 64000 Slots mit Indizes von 0 bis 63999. Auch der Tabellenspeicher erlaubt Lese- und Schreibzugriffe, aber die Zugriffsmethode ist bei diesen beiden Operationen unterschiedlich. Bevor er ausgelesen werden kann, muss zuerst ein bestimmter Tabellenspeicher-Slot mit dem Befehl **TABLE(x, wert1, wert2,…) definiert und geschrieben werden,** wobei **x** für den Index des zu definierenden Anfangs-Tabellenspeicher-Slots steht und **Wert1**, **Wert2**, ... die Werte sind, die in den Tabellenspeicher bei den Indizes x, x+1, ... geschrieben werden. Wenn er definiert und geschrieben ist, kann der Tabellenspeicher-Slot mit dem Befehl **TABLE(x)** ausgelesen werden, wobei **x** den Index des Tabellenspeicher-Slots angibt. Der Versuch, einen nicht definierten Tabellenspeicher-Slot zu lesen, hat eine Fehlermeldung zur Folge, die vom TJ1-MC__ ausgegeben wird. Der Inhalt des TABLE-Speichers wird in dem batteriegespeisten RAM-Speicher gehalten und wird auch bei abgeschalteter Versorgungsspannung nicht gelöscht. Der TABLE-Speicher bleibt auch beim Batteriewechsel erhalten, wenn dies schnell genug erfolgt.

2.3.3 Datenstrukturen und Variablen

BASIC-Programme können numerische Daten in verschiedenen Arten von Variablen speichern. Einige Variablen haben vordefinierte Funktionen, z. B. die Achsenparameter und Systemparameter; andere Variablen können vom Programmierer nach Bedarf in der Programmierung definiert werden. Die TABLE-, globalen und lokalen Variablen des TJ1-MC__ werden in diesem Abschnitt erläutert. Außerdem wird die Verwendung von Labels spezifiziert.

TABLE-Variablen

TABLE ist eine Array-Struktur, die eine Serie von Nummern enthält. Diese Nummern werden z. B. dazu verwendet, Positionen in dem Profil für einen **CAM**- oder **CAMBOX**-Befehl anzugeben. Sie können auch dazu dienen, Daten zum späteren Gebrauch zu speichern, z. B. um die Parameter zu speichern, die zum Definieren eines zu bearbeitenden Werkstücks gebraucht werden. TABLE wird von allen Tasks im TJ1-MC__ gemeinsam verwendet. Das bedeutet, dass die Werte, die von einer Task in die TABLE geschrieben werden, von anderen Task's gelesen werden können.
TABLE-Werte können mit dem Befehl **TABLE** geschrieben und gelesen werden. Die maximale Länge des Arrays beträgt 64000 Elemente, von **TABLE(0)** bis **TABLE(63999)**. Das TABLE-Array wird bis zum höchsten definierten Elemente initialisiert.

Globale Variablen

Die im VR-Speicher definierten globalen Variablen werden von allen Tasks im TJ1-MC__ gemeinsam verwendet. Das bedeutet: Wenn ein Programm, das Task 2 ausführt, VR(25) auf einen bestimmten Wert setzt, kann jedes andere Programm, das eine andere Task ausführt, den gleichen Wert aus VR(25) lesen. Dies ist zum Synchronisieren von zwei oder mehr Tasks sehr nützlich, jedoch muss darauf geachtet werden, dass nicht mehrere Programme gleichzeitig zu derselben Variablen schreiben. Der Controller hat 1024 globale Variablen, VR(0) bis VR(1023). Die Variablen werden mit dem Befehl **VR** gelesen und geschrieben.

Auf die TABLE- und VR-Daten kann aus verschiedenen laufenden Tasks zugegriffen werden. Wenn Sie die VR- oder TABLE-Variablen benutzen, stellen Sie sicher, dass nur ein Task zu einer Variablen schreiben kann. Das ist wichtig, um zu vermeiden, dass zwei Programm-Tasks unerwartet zu einer Variablen schreiben.

Lokale Variablen

Bezeichnete Variablen oder lokale Variablen können in Programmen angegeben werden und sind lokal auf die Task beschränkt. Das bedeutet, dass zwei oder mehr Programme, die verschiedene Tasks ausführen, die gleiche Variablenbezeichnung verwenden können, aber die Werte können unterschiedlich sein. Lokale Variablen können von keinem anderen Task gelesen werden, außer von dem, für den sie angeben wurden. Lokale Variablen werden immer gelöscht, wenn ein Programm gestartet wird. Die lokalen Variablen können entweder mit dem Befehl **CLEAR** oder **RESET** gelöscht werden.
Es können maximal 255 lokale Variablen angegeben werden. Nur die ersten 16 Zeichen der Bezeichnung sind Wertstellen. Nicht definierte lokale Variablen geben den Wert Null zurück. Lokale Variablen können nicht in der Befehlszeile angegeben werden.

Labels

Die BASIC-Programme werden in absteigender Reihenfolge zeilenweise ausgeführt. Labels können verwendet werden, um diese Ausführungsabfolge zu ändern; dies erfolgt mit den BASIC-Befehlen **GOTO** und **GOSUB**. Um ein Label zu definieren, muss es als erste Anweisung in einer Zeile stehen und mit einem Doppelpunkt (:) enden. Labels können Zeichenketten beliebiger Länge sein, aber nur die ersten 15 Zeichen sind Wertstellen.

Verwendung von Variablen und Labels

Jeder Task hat seine eigenen lokalen Labels und lokalen Variablen. Betrachten Sie z. B. die beiden unten gezeigten Programme:

```
start:
    FOR a = 1 bis 100
        MOVE(a)
        WAIT IDLE
    NEXT a
GOTO Start
```

```
Start:
    a=0
    REPEAT
        a = a + 1
        PRINT a
    UNTIL a = 300
GOTO Start
```

Wenn diese beiden Programme gleichzeitig in verschiedenen Tasks ausgeführt werden, haben sie ihre eigene Version der Variable **a** und des Labels **start**.
Wenn Sie Daten von zwei oder mehr Programmen gemeinsam nutzen lassen wollen, sollten VR-Variablen verwendet werden. Wenn eine große Datenmenge gespeichert werden muss, kann alternativ dazu auch der TABLE-Speicher verwendet werden.
Um ein Programm bei Verwendung einer globalen VR-Variablen besser lesbar zu machen, gibt es zwei Methoden. Die erste besteht darin, eine bezeichnete lokale Variable als Konstante in der VR-Variablen einzusetzen. Die lokale konstante Variable muss jedoch in jedem Programm angegeben werden, das die globale VR-Variable benutzt. Das mit dieser Methode geschriebene unten stehende Beispiel zeigt, wie VR(3) verwendet wird, um einen Längenparameter zu speichern, den mehrere Programme gemeinsam benutzen:

```
start:
    GOSUB initial
    VR(laenge) = x
    ...
    ...

initial:
    laenge = 3
    RETURN
```

```
start:
    GOSUB initial
    MOVE(VR(laenge))
    PRINT(VR(laenge))
    ...

initial:
    laenge = 3
    RETURN
```

Der andere Ansatz ist noch übersichtlicher und verwendet den Befehl **GLOBAL** zur Angabe des Namens als Referenz für eine der globalen VR-Variablen. Der Name kann dann von dem Programm, das die Definition für **GLOBAL** enthält und von allen anderen Programmen benutzt werden. Beachten Sie, dass das Programm mit der Definition für **GLOBAL** zuerst durchlaufen wird, bevor der Name in anderen Programmen verwendet wird. Die beste Methode besteht darin, globale Namen im Startprogramm zu definieren. Mit dieser Methode sieht das obige Beispiel so aus:

```
'Die Angabe im Startprogramm
GLOBAL laenge, 3

'In anderen Programmen nach dem Startprogramm ausgefuehrt
start:                          start:
    laenge = x                      MOVE(laenge)
    ...                             PRINT(laenge)
    ...                             ...
```

2.3.4 Mathematische Spezifikationen

Zahlenformat

Der TJ1-MC__ hat zwei Zahlenformate für numerische Werte: Gleitkommawerten mit einfacher Genauigkeit und Integer-Werte mit einfacher Genauigkeit. Gleitkommawerte mit einfacher Genauigkeit sind intern 32-Bit-Werte. Sie haben ein 8-Bit-Exponentenfeld, ein Vorzeichen-Bit und ein 23-Bit-Nachkommafeld mit einer impliziten 1 als 24. Bit. Gleitkommazahlen haben einen gültigen Bereich von $\pm 5{,}9 \times 10^{-39}$ bis $\pm 3{,}4 \times 10^{38}$.
Integer sind im Grunde Gleitkommazahlen mit dem Exponent Null.
Das bedeutet, dass die Integer-Zahlen 24 Bit umfassen. Der Integer-Bereich geht deshalb von –16777216 bis 16777215. Numerische Werte außerhalb dieses Bereichs werden als Gleitkommazahlen dargestellt.

Alle mathematischen Berechnungen erfolgen im Gleitkommaformat. Das hat zur Folge, dass bei Berechnungen von/mit größeren Werten die Genauigkeit der Ergebnisse eingeschränkt ist. Der Anwender sollte dies bei der Entwicklung der Motion Control-Anwendung berücksichtigen.

Hexadezimalformat

Der TJ1-MC__ unterstützt die Zuordnung und die Ausgabe hexadezimaler Werte. Eine Hexadezimalzahl wird eingegeben, indem der Zahl das Zeichen $ vorangestellt wird. Der gültige Bereich ist von 0x0 bis 0xFFFFFF. Beispiel:

```
>> VR(0)=$FF
>> PRINT VR(0)
255.0000
```

Mit der Funktion **HEX** kann ein Wert hexadezimal ausgegeben werden. Negative Werte werden als Zweierkomplement-Hexadezimalwert (24-Bit) dargestellt. Der gültige Bereich ist von –8388608 bis 16777215. Beispiel:

```
>> TABLE(0,-10,65536)
>> PRINT HEX(TABLE(0)),HEX(TABLE(1))
FFFFF6 10000
```

Positionierung

Bei der Positionierung rundet der TJ1-MC__ auf, wenn der berechnete Drehgeber-Flankenabstand nach dem Komma 0,9 überschreitet. Andernfalls wird der Wert nach dem Komma abgerundet. Die interne gemessene Position und die Sollposition der Achsen, die durch die Achsenparameter **MPOS** und **DPOS** dargestellt werden, haben 32-Bit-Zähler.

Gleitkomma-Vergleich

Die Vergleichsfunktion setzt kleine Unterschiede zwischen Werten gleich, um unerwartete Vergleichsergebnisse zu vermeiden. Deshalb werden zwei Werte, bei denen die Differenz weniger als $1{,}19 \times 10^{-6}$ beträgt, gleichgesetzt.

Vorrangigkeit

Die Vorrangigkeit der Operatoren ist unten angegeben:

1. Unär minus, **NOT**
2. ^
3. / *
4. **MOD**
5. + -
6. = <> > >= <= <
7. **AND OR XOR**
8. Von links nach rechts

Die beste Methode, um die Vorrangigkeit verschiedener Operatoren sicherzustellen, ist die Verwendung von Klammern.

2.4 Bewegungsausführung

Jeder Task im TJ1-MC__ hat eine Reihe von Befehlsspeichern, die die Daten aus den gegebenen Bewegungsbefehlen enthalten.

2.4.1 Profilgenerator

Der Profilgenerator hat zwei Bewegungs-Befehlsspeicher für jede Achse. Ein Befehlsspeicher mit der Bezeichnung **MTYPE** speichert die aktuelle Bewegung, d. h. die Bewegung, die zurzeit an der Achse ausgeführt wird. Der andere Befehlsspeicher mit der Bezeichnung **NTYPE** speichert die nächste Bewegung, die ausgeführt wird, nachdem die aktuelle Bewegung beendet
Eine genaue Erläuterung finden Sie in Kapitel 2.8 „Bewegungs-Befehlsspeicher“ im Trajexia Hardware-Referenzhandbuch.

Die BASIC-Programme sind vom Profilgeneratorprogramm getrennt, welches die Bewegungen der Achsen steuert.
Der Profilgenerator hat für jede Achse getrennte Funktionen, daher kann jede Achse mit ihren eigenen Achsenparametern programmiert werden (z. B. Geschwindigkeit, Beschleunigung) und sich unabhängig und simultan bewegen, oder die Achsen können mit speziellen Befehlen miteinander synchronisiert werden. Wenn ein Bewegungsbefehl verarbeitet wird, wartet der Profil-generator, bis die Bewegung beendet und der Befehlsspeicher für die erforderliche Achse leer ist; dann werden diese Befehlsspeicher mit den Daten für die nächste Bewegung geladen.

Wenn der Task- Befehlsspeicher voll ist, pausiert die Programmausführung, bis wieder Befehlsspeicher verfügbar ist. Dies gilt auch für den Befehlszeilen-Task; für diesen Zeitraum können keine Befehle gegeben werden. Trajexia Tools trennt in einem solchen Fall die Verbindung. Der Task-Parameter **PMOVE** wird auf TRUE bzw. -1 gesetzt, wenn der Task- Befehlsspeicher voll ist. Er wird auf FALSE bzw. 0 zurückgesetzt, wenn wieder Task- Befehlsspeicher verfügbar ist.

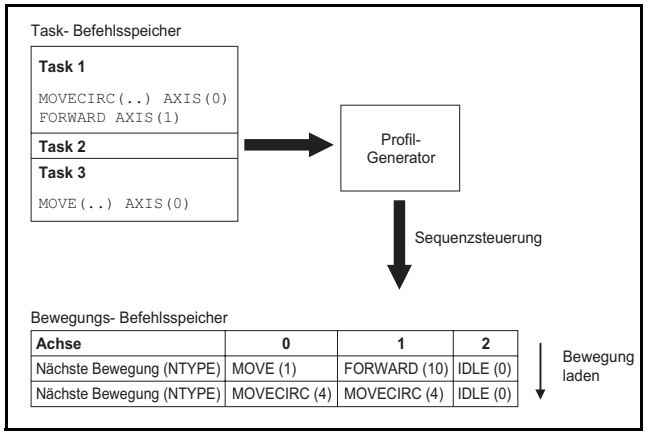
2.4.2 Sequenzsteuerung

Bei jeder Unterbrechung des Servozyklus (siehe Abschnitt 2.6.3) überprüft der Profilgenerator die **NTYPE**-Befehlsspeicher, um zu ermitteln, ob einer von ihnen verfügbar ist. Wenn einer von ihnen verfügbar ist, kontrolliert er den Task- Befehlsspeicher, um zu ermitteln, ob eine Bewegung darauf wartet, geladen zu werden. Wenn eine Bewegung geladen werden kann, werden die Daten für alle spezifizierten Achsen aus dem Task- Befehlsspeichern in den **NTYPE** – Befehlsspeicher geladen, und der entsprechenden Task-Befehlsspeicher wird als im Ruhezustand befindlich gekennzeichnet. Dieser Prozess wird als Sequenzsteuerung bezeichnet.

2.4.3 Laden eines Bewegungsbefehls

Wenn die Sequenzsteuerung abgeschlossen ist, wird der **MTYPE**-Befehlsspeicher überprüft, um zu ermitteln, ob Bewegungen geladen werden können Wenn der erforderliche **MTYPE**- Befehlsspeicher verfügbar ist, wird die Bewegung aus dem **NTYPE**- Befehlsspeicher in den **MTYPE**-Befehlsspeicher geladen, und der **NTYPE**-Befehlsspeicher wird als im Ruhezustand befindlich gekennzeichnet. Dieser Prozess wird als Bewegungsladung bezeichnet. Wenn sich eine gültige Bewegung im **MTYPE**- Befehlsspeicher befindet, wird sie verarbeitet. Wenn die Bewegung abgeschlossen ist, wird der **MTYPE**- Befehlsspeicher als im Ruhezustand befindlich gekennzeichnet.

Abb.2



2.5 Befehlszeilen-Schnittstelle

Die Befehlszeilen-Schnittstelle bietet für den Anwender eine direkte Schnittstelle zum Ausführen von Befehlen und zum Zugriff auf Parameter im System. Verwenden Sie das Terminalfenster in Trajexia Tools, wenn der TJ1-MC__ angeschlossen ist. Detaillierte Informationen hierzu finden Sie in Abschnitt 5.5.4. Der TJ1-MC__ legt die letzten 10 in der Befehlszeile gegebenen Befehle in einen Befehlsspeicher. Mit den Aufwärts- und Abwärts-Pfeiltasten kann man durch den Befehlsspeicher navigieren, um den Befehl nochmals auszuführen.

2.6 BASIC-Programme

Der TJ1-MC__ kann bis zu 14 Programme im Speicher ablegen, wenn die Kapazität des Speichers nicht überschritten wird. Der TJ1-MC__ unterstützt zum Verwalten dieser Programme einfache Dateiverwaltungsanweisungen ähnlich dem DOS-Dateisystem auf einem Computer.
Das Trajexia Tools-Softwarepaket wird verwendet, um Programme zu bearbeiten zu speichern und zu laden. Außerdem hat es verschiedene Controller-Überwachungs- und -Debugging-Funktionen. Siehe Kapitel 5.

2.6.1 Verwalten von Programmen

Trajexia Tools legt automatisch ein Projekt an, das die Programme enthält, die für eine Anwendung benutzt werden sollen. Die Programme des Projekts werden sowohl im Controller als auch im Computer gespeichert. Wenn ein Programm angelegt oder bearbeitet wird, bearbeitet Trajexia Tools beide Kopien, um stets ein genaues Backup außerhalb des Controllers bereitzustellen. Trajexia Tools stellt mit einer zyklischen Redundanzprüfung sicher, dass die beiden Versionen des Projekts identisch sind. Wenn beide unterschiedlich sind, erlaubt Trajexia Tools das Kopieren der TJ1-MC__-Version auf Datenträger oder umgekehrt.
Die Programme im Computer werden in ASCII-Textdateien gespeichert. Deshalb können Sie mit einem einfachen Texteditor ausgedruckt, bearbeitet und kopiert werden. Die Quellprogramme werden im TJ1-MC__ in einer in Tokens übersetzten Form gespeichert; daher sind die Größen der Programme im Vergleich zu den gleichen Programmen auf dem Computer im TJ1-MC__ kleiner.

Speichern von Programmen

Die Programme im TJ1-MC__ werden in dem batteriegespeisten RAM-Speicher gehalten und werden auch bei abgeschalteter Versorgungsspannung nicht gelöscht. Dies ist ähnlich wie bei den VR- und TABLE-Speichern. Der Inhalt des im RAM-Speicher gehaltenen Programms bleibt erhalten, wenn die Batterie im TJ1-MC__ installiert ist. Die Programme bleiben auch beim Batteriewechsel erhalten, wenn dieser schnell genug erfolgt. Um Programme ohne Batterie für einen längeren Zeitraum gespeichert zu halten, müssen die aktuellen Programme mit dem Befehl **EPROM** in den Flash-Speicher des Controllers kopiert und während des Hochfahrens zurückgelesen werden, was durch den Systemparameter **POWER_UP** bestimmt wird.

Programmbefehle

Der TJ1-MC__ verfügt über eine Reihe von BASIC-Befehlen, die das Anlegen, Ändern und Löschen von Programmen erlauben. Trajexia Tools bietet Schaltflächen, mit denen diese Operationen ebenfalls ausgeführt werden können, daher ist die Verwendung dieser Befehle in den Programmen normalerweise nicht erforderlich.

Befehl	Funktion
SELECT	Wählt ein Programm zum Bearbeiten, Löschen usw. aus.
NEW	Löscht das aktuell ausgewählte Programm, ein spezifiziertes Programm oder alle Programme.
DIR	Listet das Verzeichnis aller Programme auf.
COPY	Dupliziert ein spezifiziertes Programm.
RENAME	Benennt ein spezifiziertes Programm um.
DEL	Löscht das aktuell ausgewählte Programm oder ein spezifiziertes Programm.
LIST	Listet das aktuell ausgewählte Programm oder ein spezifiziertes Programm auf.

2.6.2 Programm-Kompilierung

Das TJ1-MC__-System kompiliert Programme automatisch, wenn dies erforderlich ist. Es ist normalerweise nicht erforderlich, die Kompilierung von Programmen zu erzwingen, jedoch können Programme unter dem Menüpunkt **Programm** in Trajexia Tools kompiliert werden.
Der TJ1-MC__ kompiliert Programme automatisch in folgenden Fällen:

- Das ausgewählte Programm wird kompiliert, bevor es ausgeführt wird, wenn es bearbeitet worden ist.
- Das ausgewählte Programm wird kompiliert, wenn es bearbeitet worden ist, bevor von dem ausgewählten Programm zu einem anderen Programm umgeschaltet wird.
- Das ausgewählte Programm wird mit dem Befehl **COMPILE** kompiliert.

Die Syntax und Struktur des Programms werden während der Kompilierung überprüft. Wenn die Kompilierung fehlschlägt, wird eine Meldung angezeigt, und es wird kein Programmcode generiert.
In dem Trajexia Tools-Verzeichnisfeld wird ein rotes Kreuz angezeigt.
Programme können nicht ausgeführt werden, wenn Kompilierungsfehler auftreten. Die Fehler müssen korrigiert und die Programme neu kompiliert werden.
Der Kompilierungsprozess beinhaltet auch Folgendes:

- Entfernen von Kommentaren.
- Kompilierung von Zahlen in das interne Prozessorformat.
- Konvertierung von Ausdrücken in das Postfixnotationsformat zur Ausführung.
- Vorberechnung von Variablen-Speicheradresseen.
- Berechnung und Integration von Schleifenstruktur-Zielen.

Da der Kompilierungsprozess etwas freien Speicherplatz erfordert, können unerwartete Kompilierungsfehler auftreten, wenn der freie Speicherplatz nicht ausreicht.

2.6.3 Programmausführung

Die Zeitpunkte für die Ausführung der verschiedenen Tasks und die Aktualisierung der E/A des TJ1-MC__ wiederholen sich entsprechend dem Servozyklus-Zeitraum des Systems. Der Servozyklus-Zeitraum wird durch den Systemparameter **SERVO_PERIOD** festgelegt. Der TJ1-MC__ hat einen Servozyklus-Zeitraum von 0,5, 1,0 oder 2,0 ms.

E/A-Aktualisierung

Der E/A-Zustand des TJ1-MC__ wird zu Beginn jedes Servozyklus aktualisiert.

- Der erfasste Zustand der digitalen Eingänge wird zu der System-Eingangsvariablen **IN** übertragen. Bitte beachten Sie, dass dies der Zustand ist, der im vorherigen Servozyklus erfasst wurde.
- Die analogen Ausgänge für die Drehzahlsollwerte werden aktualisiert.
- Die digitalen Eingänge werden entsprechend dem Zustand der System-Ausgangsvariablen **OP** aktualisiert.
- Der Zustand der digitalen Eingänge wird erfasst.

Bitte beachten Sie, dass keine automatische Verarbeitung der E/A-Signale erfolgt, außer zur Registrierung. Das bedeutet, dass alle Aktionen in den BASIC-Programmen programmiert werden müssen.

Wichtige Befehle

Trajexia Tools bietet verschiedene Methoden zum Ausführen, Pausieren und Stoppen von Programmen über Tasten/Schaltflächen auf der Bedienkonsole und in Bearbeitungsfenstern. Die folgenden Befehle können in der Befehlszeile zur Steuerung der Ausführung gegeben werden.

Befehl	Funktion
RUN	Führt das aktuell ausgewählte Programm oder ein spezifiziertes Programm aus, optional bei einer angegebenen Task-Nummer.
STOP	Stoppt das aktuell ausgewählte Programm oder ein spezifiziertes Programm.
HALT	Stoppt alle Programme im System.
PROCESS	Zeigt alle laufenden Tasks an.

Der Anwender kann die Priorität des Tasks festlegen, in dem das BASIC-Programm ablaufen soll. Wird ein Anwenderprogramm ohne ausdrückliche Task-Zuordnung ausgeführt, wird ihm die höchste verfügbare Task-Priorität zugewiesen.

Einstellung von Programmen, die beim Hochfahren starten sollen

Programme können so eingestellt werden, dass sie automatisch mit unterschiedlichen Prioritäten beim Einschalten der Spannungsversorgung ausgeführt werden. Nötigenfalls kann der Computer als Bedienoberfläche angeschlossen bleiben, oder er kann entfernt werden, sodass die Programme im Standalone-Betrieb laufen.
Programme können in Trajexia Tools mit der Auswahl **Autostart-Modus...** im Menü **Programm** zur automatischen Ausführung beim Hochfahren ausgewählt werden. Durch diese Auswahl wird bestimmt, welche Programme automatisch mit welcher Priorität gestartet werden. Dies kann auch mit dem BASIC-Befehl **RUNTYPE** erfolgen. Der aktuelle Zustand kann mit dem Befehl **DIR** angezeigt werden.
Weitere Informationen zu Programmsteuerung, Multitasking und Zykluszeit und finden Sie in den Abschnitten 2.2 und 2.3 des Trajexia Hardware-Referenzhandbuchs.

3 BASIC-Befehle

3.1 Kategorien

In diesem Abschnitt sind alle BASIC-Befehle nach Kategorien geordnet aufgelistet. Die Kategorien sind:

- Achsenbefehle
- Achsenparameter
- Kommunikationsbefehle und -parameter
- Konstanten
- E/A-Befehle, -funktionen und -parameter
- Mathematische Funktionen und Operationen
- Programmbefehle
- Programmsteuerungsbefehle
- Slot-Parameter and -modifikatoren
- Systembefehle und -funktionen
- Systemparameter
- Task-Befehle und -parameter

Die Listen dienen nur zur Schnellübersicht. Eine vollständige Beschreibung der Befehle finden Sie in alphabetischer Reihenfolge im nächsten Abschnitt.

3.1.1 Achsenbefehle

Bezeichnung	Beschreibung
ACC	Ändert **ACCEL** und **DECEL** gleichzeitig.
ADD_DAC	Summe zu dem **DAC**-Wert einer Achse zum analogen Ausgang der Basisachse.
ADDAX	Stellt einen Link zu einer überlagernden Achse her. Alle gewünschten Positionsbewegungen für die überlagernde Achse werden zu allen Bewegungen hinzugefügt, die aktuell ausgeführt werden.
B_SPLINE	Erweitert das im **TABLE** -Speicher gespeicherte Profil mit der mathematischen B-Spline-Funktion.
BASE	Dient zur Bestimmung der Basisachse, auf welche die Befehle und Parameter angewendet werden.
CAM	Bewegt eine Achse gemäß den Werten eines Bewegungsprofils, das im TABLE-Variablen-Array gespeichert ist.
CAMBOX	Bewegt eine Achse gemäß den Werten eines Bewegungsprofils, das im TABLE-Variablen-Array gespeichert ist. Die Bewegung ist mit der gemessenen Bewegung einer anderen Achse synchronisiert, um ein ständig variables Software-Getriebe zu bilden.
CANCEL	Bricht die Bewegung einer Achse ab.
CONNECT	Verbindet die Sollposition einer Achse mit den gemessenen Bewegungen der Achse, die für **Antriebs_achse** spezifiziert sind, um ein elektronisches Getriebe zu erzeugen.
DATUM	Führt eine von 7 Nullpunkt-Suchsequenzen durch, um eine Achse auf eine Absolutposition zu setzen oder einen Bewegungsfehler zurückzusetzen.
DEFPOS	Definiert die aktuelle Position als neue Absolutposition.
DISABLE_GROUP	Gruppiert Achsen zur Fehler-Deaktivierung zusammen.
DRIVE_ALARM	Überwacht den aktuellen Alarm.
DRIVE_CLEAR	Löscht den Alarmzustand des Servotreibers.
DRIVE_READ	Liest die spezifizierten Parameter des Servotreibers.
DRIVE_RESET	Setzt den Servotreiber zurück.
DRIVE_WRITE	Schreibt einen spezifischen Wert zu dem spezifizierten Parameter des Servotreibers.
ENCODER_READ	Liest einen Parameter des EnDat-Absolutwert-Drehgebers.
ENCODER_WRITE	Schreibt zu einem Parameter des EnDat-Absolutwert-Drehgebers.
FORWARD	Bewegt eine Achse kontinuierlich mit der Geschwindigkeit vorwärts, die im Parameter **SPEED** eingestellt ist.
HW_PSWITCH	Stellt den Hardware-Schalter an Ausgang 0 des TJ1-FL02 auf Ein oder Aus, wenn vordefinierte Positionen erreicht sind.
MECHATROLINK	Initialisiert den MECHATROLINK-II-Bus und führt verschiedene Operationen bei MECHATROLINK-II-Stationen aus, die an den Bus angeschlossen sind.
MHELICAL	Interpoliert 3 orthogonale Achsen in einer Wendelbewegung.

Bezeichnung	Beschreibung
MOVE	Bewegt eine oder mehrere Achsen mit der gewünschten Geschwindigkeit, Beschleunigung und Verzögerung zu der Position, die als Inkrement von der aktuellen Position spezifiziert ist.
MOVEABS	Bewegt eine oder mehrere Achsen mit der gewünschten Geschwindigkeit, Beschleunigung und Verzögerung zu der Position, die als Absolutposition spezifiziert ist.
MOVECIRC	Interpoliert 2 orthogonale Achsen in einem Kreisbogen.
MOVELINK	Erzeugt eine lineare Bewegung der Basisachse, die über ein Software-Getriebe mit der gemessenen Position einer synchronisierten Achse verbunden ist.
MOVEMODIFY	Ändert die absolute Endposition der aktuellen einachsigen linearen Bewegung (**MOVE** oder **MOVEABS**).
RAPIDSTOP	Bricht die aktuelle Bewegung aller Achsen ab.
REGIST	Erfasst eine Achsenposition, wenn ein Druckmarkeneingang oder die „Z“-Markierung beim Drehgeber erkannt wird.
REVERSE	Bewegt eine Achse kontinuierlich mit der Geschwindigkeit rückwärts, die im Parameter **SPEED** eingestellt ist.
STEP_RATIO	Stellt das Verhältnis für den Achsen-Schrittmotorausgang ein.

3.1.2 Achsenparameter

Bezeichnung	Beschreibung
ACCEL	Enthält die Beschleunigungsrate der Achse.
ADDAX_AXIS	Enthält die Anzahl der Achsen, mit denen die Basisachse aktuell durch **ADDAX** synchronisiert ist.
ATYPE	Enthält den Achsentyp.
AXIS_DISPLAY	Wählt Informationen aus, die durch die LEDs auf der Frontabdeckung des TJ1-FL02 dargestellt werden.
AXIS_ENABLE	Aktiviert und deaktiviert bestimmte Achsen unabhängig von anderen Achsen.
AXISSTATUS	Enthält den Achsenzustand.
CLOSE_WIN	Definiert das Ende des Fensters, in welchem eine Druckmarke erwartet wird.
CLUTCH_RATE	Definiert die Geschwindigkeit des Verbindungsaufbaus, wenn der Befehl **CONNECT** verwendet wird.
CREEP	Enthält die Schleichganggeschwindigkeit.
D_GAIN	Enthält die Differenzialverstärkung.
DAC_SCALE	Stellt den Maßstab und die Polarität ein, die für **DAC**-Werte gelten.
DATUM_IN	Enthält die Eingangsnummer, die als Nullpunkteingang verwendet werden soll.
DECEL	Enthält die Verzögerungsrate der Achse.
DEMAND_EDGES	Enthält den aktuellen Wert des Achsenparameters **DPOS** in Drehgeber-Flanken.
DPOS	Enthält die Sollposition, die von den Bewegungsbefehlen generiert wurde.
DRIVE_CONTROL	Wählt Daten zur Überwachung mit **DRIVE_MONITOR** für Achsen aus, die über den MECHATROLINK-II-Bus verbunden sind. Bei Achsen, die über den TJ1-FL02 verbunden sind, stellt **DRIVE_CONTROL** die Ausgänge des TJ1-FL02 ein.
DRIVE_INPUTS	Speichert E/A-Daten des Treibers, der mit dem MECHATROLINK-II-Bus verbunden ist. Die Daten werden bei jedem Servozyklus aktualisiert.
DRIVE_MONITOR	Überwacht Daten des Servotreibers, der mit dem MECHATROLINK-II-Bus verbunden ist. Die Daten werden bei jedem Servozyklus aktualisiert.
DRIVE_STATUS	Enthält den aktuellen Zustand des Servotreibers.
ENCODER	Enthält eine Grobkopie des Drehgeber-Hardwareregisters.
ENCODER_BITS	Stellt die Bitanzahl des Absolutwert-Drehgebers ein, der an den TJ1-FL02 angeschlossen ist.
ENCODER_CONTROL	Steuert die Betriebsart des EnDat Absolutwert-Drehgebers.
ENCODER_ID	Gibt den ID-Wert des Absolutwert-Drehgebers zurück, der an den TJ1-FL02 angeschlossen ist.
ENCODER_RATIO	Stellt Skalierungswerte für eingehende Drehgeberzähler ein.
ENCODER_STATUS	Gibt den Zustand des Tamagawa-Absolutwert-Drehgebers zurück.
ENCODER_TURNS	Gibt den Multi-Umdrehungszähler des Absolutwert-Drehgebers zurück.

Bezeichnung	Beschreibung
ENDMOVE	Speichert die Endposition der aktuellen Bewegung.
ERRORMASK	Enthält den Maskenwert, der bestimmt, ob **MOTION_ERROR** abhängig vom Achsenzustand auftritt.
FAST_JOG	Enthält die Eingangsnummer, die als Schnelltippeingang verwendet werden soll.
FASTDEC	Definiert das Verzögerungsverhältnis „Rampe bis Null“, wenn ein Achsen-Begrenzungsschalter oder eine Grenzposition erreicht ist.
FE	Enthält den Schleppfehler (FE).
FE_LATCH	Enthält den Wert des Schleppfehlers, der den **MOTION_ERROR**- ausgelöst hat.
FE_LIMIT	Enthält den maximal zulässigen Schleppfehler.
FE_LIMIT_MODE	Definiert, wie sich der Schleppfehler auf den **MOTION_ERROR**-Zustand auswirkt.
FE_RANGE	Enthält die Warnbereichsgrenze für den Schleppfehler.
FHOLD_IN	Enthält die Eingangsnummer, die als Feedhold-Eingang verwendet werden soll.
FHSPEED	Enthält die Feedhold-Drehzahl.
FS_LIMIT	Enthält die Absolutposition der Vorwärts-Softwarebegrenzung.
FWD_IN	Enthält die Eingangsnummer, die als Vorwärts-Begrenzungseingang verwendet werden soll.
FWD_JOG	Enthält die Eingangsnummer, die als Vorwärtstippeingang verwendet werden soll.
I_GAIN	Enthält die Integrationsregelungs-Verstärkung.
INVERT_STEP	Schaltet eine Hardware-Invertierung in den Ausgangsschaltkreis ein.
JOGSPEED	Bestimmt die Schrittdrehzahl.
LINKAX	Enthält die Achsennummer der synchronisierten Achsen während einer synchronisierten Bewegung.
MARK	Erkennt das primäre Druckmarkenereignis bei einem Druckmarkeneingang.
MARKB	Erkennt das sekundäre Druckmarkenereignis bei einem Druckmarkeneingang.

Bezeichnung	Beschreibung
MERGE	Ist ein Software-Schalter, der zum Aktivieren oder Deaktivieren der Verknüpfung aufeinander folgender Bewegungen verwendet werden kann.
MPOS	Ist die Position der Achse, wie sie vom Drehgeber gemessen wurde.
MSPEED	Stellt die Veränderung in der gemessenen Position beim letzten Servozyklus dar.
MTYPE	Enthält die Art der Bewegung, die aktuell ausgeführt wird.
NTYPE	Enthält die Art der Bewegung im Befehlsspeicher „Nächste Bewegung“.
OFFPOS	Enthält einen Offset, der auf die Sollposition angewendet wird, ohne die Bewegung in anderer Weise zu beeinflussen.
OPEN_WIN	Definiert den Anfang des Fensters, in welchem eine Druckmarke erwartet wird.
OUTLIMIT	Enthält den Grenzwert, der den Drehzahlsollwert-Ausgang des TJ1-MC__ begrenzt.
OV_GAIN	Enthält die Verstärkung für die Ausgangsgeschwindigkeitsregelung.
P_GAIN	Enthält die Proportionalregelungsverstärkung.
REG_POS	Enthält die Position, an der ein Druckmarkenereignis aufgetreten ist.
REG_POSB	Enthält die Position, an der das sekundäre Druckmarkenereignis aufgetreten ist.
REMAIN	Ist die verbleibende Entfernung bis zum Ende der aktuellen Bewegung.
REMOTE_ERROR	Gibt bei einem MECHATROLINK-II-Anschluss des Servotreibers die Anzahl der Fehler zurück.
REP_DIST	Enthält den Wiederholungsabstand oder stellt diesen ein.
REP_OPTION	Steuert die Anwendung des Achsenparameters REP_DIST.
REV_IN	Enthält die Eingangsnummer, die als Rückwärts-Begrenzungseingang verwendet werden soll.
REV_JOG	Enthält die Eingangsnummer, die als Rückwärtstippeingang verwendet werden soll.
RS_LIMIT	Enthält die Absolutposition der Rückwärts-Softwarebegrenzung.

Bezeichnung	Beschreibung
S_REF	Enthält den Wert für den Drehzahlsollwert, der angewendet wird, wenn sich die Achse im offenen Regelkreis befindet.
S_REF_OUT	Enthält den Wert für den Drehzahlsollwert, der sowohl bei offenem als auch bei geschlossenem Regelkreis auf den Servotreiber angewendet wird.
SERVO	Ermittelt, ob die Achse mit oder ohne Lageregelung betrieben wird.
SPEED	Enthält die gewünschte Geschwindigkeit in Einheiten.
SRAMP	Enthält den S-Kurvenfaktor.
T_REF	Enthält den Wert des Drehmoments, für den Servomotor.
TRANS_DPOS	Enthält die gewünschte Achsenposition beim Ausgang einer Rahmentransformation.
UNITS	Enthält den Einheiten-Umwandlungsfaktor.
VERIFY	Wählt zwischen unterschiedlichen Betriebsarten bei einer Schrittmotor-Ausgangsachse aus.
VFF_GAIN	Enthält die Verstärkung für die Drehzahl-Vorsteuerungsregelung.
VP_SPEED	Enthält die Drehzahlprofil-Geschwindigkeit.

3.1.3 Kommunikationsbefehle und -parameter

Bezeichnung	Beschreibung
FINS_COMMS	Sendet FINS-Speicherlese- und Speicherschreibbefehle an eine bestimmte FINS-Servereinheit.
HLM_COMMAND	Führt einen spezifischen Host-Link-Befehl an den Slave aus.
HLM_READ	Liest Daten aus dem Host-Link-Slave zum VR- oder TABLE-Variablen-Array.
HLM_STATUS	Stellt den Zustand des letzten Host-Link-Masterbefehls dar.
HLM_TIMEOUT	Definiert die Host Link-Master-Zeitspanne für die Zeitüberschreitung.
HLM_WRITE	Schreibt Daten aus dem VR- oder TABLE-Variablen-Array zum Host-Link-Slave.
HLS_NODE	Definiert die Slave-Einheitennummer für das Host-Link Slave-Protokoll.
SETCOM	Stellt die serielle Kommunikation ein.

3.1.4 Konstanten

Bezeichnung	Beschreibung
FALSE	Ist gleich dem numerischen Wert 0.
OFF	Ist gleich dem numerischen Wert 0.
ON	Ist gleich dem numerischen Wert 1.
PI	Ist gleich dem numerischen Wert 3,1416.
TRUE	Ist gleich dem numerischen Wert –1.

3.1.5 E/A-Befehle, -funktionen und -parameter

Bezeichnung	Beschreibung
GET	Wartet auf die Ankunft eines einzelnen Zeichens und ordnet den ASCII-Code des Zeichens einer Variablen zu.
IN	Gibt den Wert digitaler Eingänge zurück.
INDEVICE	Der Parameter definiert das Standard-Eingabegerät.
INPUT	Wartet auf den Empfang einer Zeichenkette und ordnet den numerischen Wert einer Variablen zu.
KEY	Gibt TRUE oder FALSE zurück, je nachdem, ob ein Zeichen empfangen wurde.
LINPUT	Wartet auf eine Zeichenkette und überträgt sie in VR-Variablen.
OP	Stellt einen oder mehrere Ausgänge ein oder gibt den Zustand der ersten 24 Ausgänge zurück.
OUTDEVICE	Definiert das Standard-Ausgabegerät.
PRINT	Gibt eine Zeichenserie an eine serielle Schnittstelle aus.
PSWITCH	Schaltet einen Ausgang ein, wenn eine vordefinierte Position erreicht ist und schaltet ihn aus, wenn eine zweite Position erreicht ist.

3.1.6 Mathematische Funktionen und Operanden

Bezeichnung	Beschreibung
+ (ADDITION)	Addiert zwei Ausdrücke.
– (SUBTRAKTION)	Subtrahiert zwei Ausdrücke.
*** (MULTIPLIKATION)**	Multipliziert zwei Ausdrücke.
/ (DIVISION)	Dividiert zwei Ausdrücke.
^ (POTENZ)	Überträgt die Potenz eines Ausdrucks auf den anderen Ausdruck.
= (IST GLEICH)	Überprüft zwei Ausdrücke, um zu ermitteln, ob sie gleich sind.
= (ZUORDNUNG)	Ordnet einen Ausdruck einer Variablen zu.
<> (IST UNGLEICH)	Überprüft zwei Ausdrücke, um zu ermitteln, ob sie ungleich sind.
> (IST GRÖßER ALS)	Überprüft zwei Ausdrücke, um zu ermitteln, ob der Ausdruck auf der linken Seite größer ist als der Ausdruck auf der rechten Seite.
>= (IST GRÖßER ODER GLEICH)	Überprüft zwei Ausdrücke, um zu ermitteln, ob der Ausdruck auf der linken Seite größer als oder gleich der/dem Ausdruck auf der rechten Seite ist.
< (IST KLEINER ALS)	Überprüft zwei Ausdrücke, um zu ermitteln, ob der Ausdruck auf der linken Seite kleiner ist als der Ausdruck auf der rechten Seite.
<= (IST KLEINER ODER GLEICH)	Überprüft zwei Ausdrücke, um zu ermitteln, ob der Ausdruck auf der linken Seite kleiner als oder gleich der/dem Ausdruck auf der rechten Seite ist.
ABS	Gibt den Absolutwert eines Ausdrucks zurück.
ACOS	Gibt den Arkuskosinus eines Ausdrucks zurück.
AND	Führt eine UND-Verknüpfung bei übereinstimmenden Bits der Integer-Teile von zwei Ausdrücken durch.
ASIN	Gibt den Arkussinus eines Ausdrucks zurück.
ATAN	Gibt den Arkustangens eines Ausdrucks zurück.
ATAN2	Gibt den Arkustangens der komplexen Zahl zurück, die durch zwei Ausdrücke erzeugt wird und ungleich Null ist.
COS	Gibt den Kosinus eines Ausdrucks zurück.
EXP	Gibt den Exponentialwert eines Ausdrucks zurück.
FRAC	Gibt den Bruchteil eines Ausdrucks zurück.
IEEE_IN	Gibt Gleitkommazahlen im Format IEEE zurück, dargestellt durch vier Byte.
IEEE_OUT	Gibt ein einzelnes Byte zurück, das aus der Gleitkommazahl im Format IEEE extrahiert ist.
INT	Gibt den Integer-Teil eines Ausdrucks zurück.
LN	Gibt den natürlichen Logarithmus eines Ausdrucks zurück.
MOD	Gibt den Modulus von zwei Ausdrücken zurück.
NOT	Führt eine Negation bei übereinstimmenden Bits des Integer-Teils von zwei Ausdrücken durch.
OR	Führt eine ODER-Verknüpfung zwischen übereinstimmenden Bits der Integer-Teile von zwei Ausdrücken durch.
SGN	Gibt das Vorzeichen eines Ausdrucks zurück.
SIN	Gibt den Sinus eines Ausdrucks zurück.
SQR	Gibt die Quadratwurzel eines Ausdrucks zurück.
TAN	Gibt den Tangens eines Ausdrucks zurück.
XOR	Führt eine XOR-Verknüpfung zwischen übereinstimmenden Bits der Integer-Teile von zwei Ausdrücken durch.

3.1.7 Programmbefehle

Bezeichnung	Beschreibung
' (KOMMENTARFELD)	Legt fest, dass eine Zeile nicht ausgeführt wird.
: (ANWEISUNGS-TRENNZEICHEN)	Erlaubt mehr Anweisungen in einer Zeile.
AUTORUN	Startet alle Programme, die beim Hochfahren ausgeführt werden sollen.
COMPILE	Kompiliert das aktuelle Programm.
COPY	Kopiert ein im Motion-Controller bestehendes Programm zu einem neuen Programm.

Bezeichnung	Beschreibung
DEL	Löscht ein Programm aus dem Motion-Controller.
DIR	Zeigt eine Liste von Programmen im Motion-Controller unter Angabe von Größe und RUNTYPE am Standardausgang an.
EDIT	Erlaubt die Modifizierung eines Programms über ein VT100-Terminal.
EPROM	Speichert ein Programm im Flash-Speicher.
LIST	Druckt das Programm auf dem Standard-Ausgabegerät aus.
NEW	Löscht alle Zeilen eines Programms im Motion-Controller.
PROCESS	Gibt den Ablaufzustand und die Task-Nummer für jeden aktuellen Task zurück.
RENAME	Ändert den Namen eines Programms im Motion-Controller.
RUN	Führt ein Programm aus.
RUNTYPE	Ermittelt, ob ein Programm beim Hochfahren ausgeführt wird und in welchem Task es laufen soll.
SELECT	Spezifiziert das aktuelle Programm.
STEPLINE	Führt eine einzelne Zeile in einem Programm aus.
STOP	Stoppt die Programmausführung.
TROFF	Hebt eine Aufzeichnung bei der aktuellen Zeile auf und nimmt die normale Programmausführung wieder auf.
TRON	Erzeugt einen Haltepunkt in einem Programm.

3.1.8 Programmsteuerungsbefehle

Bezeichnung	Beschreibung
FOR..TO..STEP..NEXT	Eine Schleife erlaubt es, ein Programmsegment mit ansteigender/absteigender Variable zu wiederholen.
GOSUB..RETURN	Springt zu einer Subroutine in der Zeile hinter dem Label. Die Programmausführung kehrt zur nächsten Anweisung zurück, wenn ein „RETURN“ auf Seite 132 gegeben wird.
GOTO	Springt zu der Zeile, die das Label enthält.
IF..THEN..ELSE..ENDIF	Steuert den Programmablauf basierend auf den Ergebnissen der Bedingung.
ON.. GOSUB oder **ON.. GOTO**	Ermöglicht einen bedingten Sprung zu einem von mehreren Labels.
REPEAT..UNTIL	Die Schleife erlaubt es, das Programmsegment zu wiederholen, bis die Bedingung „TRUE“ auf Seite 146 wird.
WHILE..WEND	Die Schleife erlaubt es, das Programmsegment zu wiederholen, bis die Bedingung **FALSE** wird.

3.1.9 Slot-Parameter und -modifikatoren

Bezeichnung	Beschreibung
COMMSTYPE	Enthält den Baugruppentyp in einem Controller-Slot.
FPGA_VERSION	Gibt mit **Baugruppen_nummer** die FPGA-Version in einem Controller-System zurück.
SLOT	Ist ein Modifikator, der die Slot-Nummer der Baugruppe spezifiziert.

3.1.10 Systembefehle und -funktionen

Bezeichnung	Beschreibung
$ (HEXADEZIMALE EINGABE)	Ordnet einer Variablen eine Hexadezimalzahl zu.
AXIS	Stellt die Achse für einen Befehl, das Lesen eines Achsenparameters oder die Zuordnung zu einer bestimmten Achse ein.
BASICERROR	Wird verwendet, um eine bestimmte Routine zu starten, wenn ein Fehler in einem BASIC-Befehl auftritt.
CLEAR	Löscht alle globalen Variablen und die lokalen Variablen des aktuellen Tasks.
CLEAR_BIT	Löscht ein Bit einer VR-Variablen.
CLEAR_PARAMS	Setzt alle im Flash-EPROM gespeicherten Parameter und Variablen auf ihre Standardwerte zurück.

Bezeichnung	Beschreibung
CONSTANT	Legt eine Konstante zur Verwendung in einem BASIC-Programm fest.
DATE$	Druckt das aktuelle Datum als Zeichenkette.
DAY$	Druckt den aktuellen Tag als Zeichenkette.
DEVICENET	Konfiguriert die TJ1-DRT (DeviceNet-Slave-Baugruppe) für den Datenaustausch oder gibt den Datenaustausch-Zustand der TJ1-DRT zurück.
ETHERNET	Liest und setzt verschiedene Parameter der TJ1-MC__ Ethernet-Schnittstelle.
EX	Setzt den Controller zurück.
FLAG	Setzt und liest eine Bank von 32 Bit.
FLAGS	Liest und setzt FLAGS als Block.
FREE	Gibt die Größe des verfügbaren Speichers zurück.
GLOBAL	Legt einen Referenzwert für eine der VR-Variablen fest.
HALT	Stoppt die Ausführung aller aktuell laufenden Programme.
INITIALISE	Setzt alle Achsen und Parameter auf ihre Standardwerte zurück.
INVERT_IN	Kehrt die Eingangskanäle 0–31 in der Software um.
INVERTER_COMMAND	Liest E/A und löscht Alarme des Frequenzumrichters.
INVERTER_READ	Liest Parameter, Alarme, Drehzahl- und Drehmomentsollwerte des Frequenzumrichters.
INVERTER_WRITE	Schreibt Parameter, Alarme, Drehzahl- und Drehmomentsollwerte des Frequenzumrichters.
LIST_GLOBAL	Zeigt alle GLOBAL- und CONSTANT-Variablen an.
LOCK	Verhindert, dass Programme angezeigt oder modifiziert werden.
PROFIBUS	Konfiguriert die TJ1-PRT (PROFIBUS-DP-Slave-Baugruppe) zum Austausch von E/A-Daten mit dem Master und gibt den Zustand der TJ1-PRT zurück.
READ_BIT	Gibt den Wert des spezifizierten Bits der spezifizierten VR-Variablen zurück.
RESET	Setzt alle lokalen Variablen eines Tasks zurück.
SCOPE	Programmiert das System so, dass bei jedem Abtastintervall automatisch bis zu vier Parameter im TABLE-Variablen-Array gespeichert werden.
SET_BIT	Setzt ein Bit in einer VR-Variablen auf eins.
TABLE	Schreibt und liest Daten zu und aus dem TABLE-Variablen-Array.
TABLEVALUES	Gibt eine Werteliste aus dem TABLE-Speicher zurück.
TIME$	Druckt die aktuelle Zeit als Zeichenkette.
TRIGGER	Startet einen zuvor gesetzten **SCOPE**-Befehl.
VR	Schreibt und liest Daten zu und aus den globalen (VR-) Variablen.
VRSTRING	Kombiniert VR-Speicherwerte, sodass sie als Zeichenkette gedruckt werden können.
WA	Pausiert die Programmausführung für die angegebene Anzahl an Millisekunden.
WAIT IDLE	Setzt die Programmausführung aus, bis die Basisachse ihre aktuelle Bewegung und eventuelle weitere im Befehlsspeicher befindliche Bewegungen ausgeführt hat.
WAIT LOADED	Setzt die Programmausführung aus, bis die Basisachse keine weiteren Bewegungen außer der aktuell ausgeführten Bewegung mehr im Befehlsspeicher hat.
WAIT UNTIL	Wertet die Bedingung wiederholt aus, bis sie **TRUE** ist.

3.1.11 Systemparameter

Bezeichnung	Beschreibung
AIN	Speichert den Wert des analogen Eingangskanals.
AOUT	Speichert den Wert des analogen Ausgangskanals.
BATTERY_LOW	Gibt den aktuellen Zustand der Batterie zurück.
CHECKSUM	Enthält die Prüfsumme für die Programme im RAM-Speicher.
COMMSERROR	Enthält alle Kommunikationsfehler, die seit der letzten Initialisierung aufgetreten sind.
CONTROL	Enthält den an das System angeschlossenen Typ des TJ1-MC__.
D_ZONE_MAX	Steuert den DAC-Ausgang in Verbindung mit dem Schleppfehler-Wert.

Bezeichnung	Beschreibung
D_ZONE_MIN	Steuert den DAC-Ausgang in Verbindung mit dem Schleppfehler-Wert.
DATE	Gibt das aktuelle Datum der Echtzeit-Uhr zurück oder stellt dieses ein.
DAY	Gibt den aktuellen Tag zurück oder stellt diesen ein.
DISPLAY	Legt fest, welche E/A-Kanäle durch die LEDs auf der Vorderseite angezeigt werden.
ERROR_AXIS	Enthält die Nummer der Achse, die den Bewegungsfehler verursacht hat.
FRAME	Spezifiziert den Operationsrahmen für Rahmentransformationen.
LAST_AXIS	Enthält die Nummer der letzten Achse, die vom System bearbeitet wurde.
MOTION_ERROR	Enthält einen Fehlermerker für Achsenbewegungsfehler.
NAIO	Gibt die Anzahl der analogen Kanäle zurück, die an den MECHATROLINK-II-Bus angeschlossen sind.
NEG_OFFSET	Wendet einen negativen Offset auf das DAC-Signal vom Servo-Regelkreis an.
NIO	Enthält die Anzahl der Ein- und Ausgänge, die an das System angeschlossen sind.
POWER_UP	Ermittelt, ob Programme beim Hochfahren oder Zurücksetzen aus dem Flash-EPROM gelesen werden sollen.
POS_OFFSET	Wendet einen positiven Offset auf das DAC-Signal vom Servo-Regelkreis an.
SCOPE_POS	Enthält die aktuelleTABLE-Position, an der der SCOPE-Befehl zurzeit den ersten Parameter speichert.
SERVO_PERIOD	Stellt den Servozyklus-Zeitraum des TJ1-MC__ ein.
SYSTEM_ERROR	Enthält die Systemfehler seit der letzten Initialisierung.
TIME	Gibt die aktuelle Zeit der Echtzeit-Uhr zurück.
TSIZE	Gibt die aktuelle Größe der zurzeit definierten Tabelle zurück.
VERSION	Gibt die Versionsnummer der Controller-Firmware zurück.
WDOG	Software-Schalter, der die Servotreiber aktiviert.

3.1.12 Task-Befehle und -parameter

Bezeichnung	Beschreibung
ERROR_LINE	Enthält die Nummer der Zeile, die den letzten BASIC-Programmfehler verursacht hat.
PMOVE	Enthält den Zustand der Task- Befehlsspeicher.
PROC	Erlaubt den Zugriff auf einen Prozessparameter aus einem bestimmten Prozess.
PROC_STATUS	Gibt den Zustand des spezifizierten Prozesses zurück.
PROCNUMBER	Enthält die Nummer des Tasks, in dem das aktuelle ausgewählte Programm läuft.
RUN_ERROR	Enthält die Nummer des letzten BASIC-Fehlers, der im angegebenen Task aufgetreten ist.
TICKS	Enthält den aktuellen Zähler des Task-Taktimpulses.

3.2 Alle BASIC-Befehle

3.2.1 + (Addition)

Typ	Mathematische Funktion
Syntax	**ausdruck1 + ausdruck2**
Beschreibung	Der Operator **+** addiert zwei Ausdrücke.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**Ergebnis = 4 + 3** Ordnet der Variablen **Ergebnis** den Wert 7 zu.
Siehe auch	–

3.2.2 – (Subtraktion)

Typ	Mathematische Funktion
Syntax	**ausdruck1 – ausdruck2**
Beschreibung	Der Operator **–** subtrahiert **ausdruck2** von **ausdruck1**.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**Ergebnis = 10 – 2** Ordnet der Variablen **Ergebnis** den Wert 8 zu.
Siehe auch	–

3.2.3 * (Multiplikation)

Typ	Mathematische Funktion
Syntax	**ausdruck1 * ausdruck2**
Beschreibung	Der Operator ***** multipliziert zwei Ausdrücke.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**Ergebnis = 3 * 7** Ordnet der Variablen **Ergebnis** den Wert 21 zu.
Siehe auch	–

3.2.4 / (Division)

Typ	Mathematische Funktion
Syntax	**ausdruck1/ausdruck2**
Beschreibung	Der Operator/dividiert **ausdruck1** durch **ausdruck2**.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**Ergebnis = 11/4** Ordnet der Variablen **Ergebnis** den Wert 2,75 zu.
Siehe auch	–

3.2.5 ^ (Potenz)

Typ	Mathematische Funktion
Syntax	**ausdruck_1 ^ ausdruck_2**
Beschreibung	Der Operator **^** potenziert die Basis **ausdruck_1** mit dem Exponenten **ausdruck_2**. Bei dieser Operation werden Gleitkomma-Algorithmen verwendet, wodurch kleine Abweichungen bei Integer-Berechnungen auftreten können.
Argumente	• **ausdruck_1** Ein BASIC-Ausdruck. • **ausdruck_2** Ein BASIC-Ausdruck.
Beispiel	**Ergebnis = 2 ^ 5** Ordnet der Variablen **Ergebnis** den Wert 32 zu.
Siehe auch	–

3.2.6 = (Ist gleich)

Typ	Mathematische Funktion
Syntax	**ausdruck1 = ausdruck2**
Beschreibung	Der Operator **=** gibt **TRUE** zurück, wenn **ausdruck1** gleich **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a = 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der gleich 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.7 = (Zuordnung)

Typ	Mathematische Funktion
Syntax	**variable = ausdruck**
Beschreibung	Der Operator **=** ordnet den Wert des Ausdrucks der Variablen zu.
Argumente	• **variable** Ein Variablenname. • **ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**var = 18** Ordnet der Variablen **var** den Wert 18 zu.
Siehe auch	–

3.2.8 <> (Ist ungleich)

Typ	Mathematische Funktion
Syntax	**ausdruck1 <> ausdruck2**
Beschreibung	Der Operator **<>** gibt **TRUE** zurück, wenn **ausdruck1** ungleich **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a <> 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der ungleich 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.9 > (Ist größer als)

Typ	Mathematische Funktion
Syntax	**ausdruck1 > ausdruck2**
Beschreibung	Der Operator **>** gibt **TRUE** zurück, wenn **ausdruck1** größer als **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a > 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der größer als 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.10 >= (Ist größer oder gleich)

Typ	Mathematische Funktion
Syntax	**ausdruck1 >= ausdruck2**
Beschreibung	Der Operator **>=** gibt **TRUE** zurück, wenn **ausdruck1** größer oder gleich **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a >= 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der größer oder gleich 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.11 < (Ist kleiner als)

Typ	Mathematische Funktion
Syntax	**ausdruck1 < ausdruck2**
Beschreibung	Der Operator **<** gibt **TRUE** zurück, wenn **ausdruck1** kleiner als **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a < 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der kleiner als 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.12 <= (Ist kleiner oder gleich)

Typ	Mathematische Funktion
Syntax	**ausdruck1 <= ausdruck2**
Beschreibung	Der Operator **<=** gibt **TRUE** zurück, wenn **ausdruck1** kleiner oder gleich **ausdruck2** ist, andernfalls gibt er **FALSE** zurück.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF a <= 10 THEN GOTO label1** Wenn die Variable **a** einen Wert enthält, der kleiner oder gleich 10 ist, fährt die Programmausführung bei dem Label **label1** fort. Andernfalls fährt die Programmausführung mit der nächsten Anweisung fort.
Siehe auch	–

3.2.13 $ (Hexadezimale Eingabe)

Typ	Systembefehl
Syntax	**$hex_num**
Beschreibung	Der Befehl **$** macht die nachfolgende Zahl zu einer Hexadezimalzahl.
Argumente	• **hex_num** Eine Hexadezimalzahl (bestehend aus den Zeichen 0–9 und A–F). **hex_num** gilt für den Bereich von 0 bis FFFFFF.
Beispiel	**>>TABLE(0,$F,$ABCD)** **>>TABLE(0),TABLE(1) drucken** **15.0000 43981.0000**
Siehe auch	**HEX (PRINT)**

3.2.14 ' (Kommentarfeld)

Typ	Programmbefehl
Syntax	**'**
Beschreibung	**'** kennzeichnet alle folgenden Zeichen in einer Zeile als Kommentare und nicht als Programmcode. Ein Kommentar wird nicht ausgeführt, wenn das Programm läuft. **'** kann am Anfang einer Zeile oder nach einer gültigen Anweisung gesetzt werden.
Argumente	–
Beispiel	**' Diese Zeile wird nicht gedruckt** **PRINT "Start"**
Siehe auch	–

3.2.15 : (Anweisungs-Trennzeichen)

Typ	Programmbefehl
Syntax	**:**
Beschreibung	Das Anweisungs-Trennzeichen **:** trennt mehrere BASIC-Anweisungen in einer Zeile. Es kann in der Befehlszeile und in Programmen verwendet werden.
Argumente	–
Beispiel	**PRINT "DIESE ZEILE": GET niedrig : PRINT "bewirkt drei Dinge"**
Siehe auch	–

3.2.16

Typ	Sonderzeichen
Syntax	**#**
Beschreibung	Das Symbol **#** wird verwendet, um einen Kommunikationskanal zu spezifizieren, der für serielle Eingangs-/Ausgangsbefehle benutzt wird. Hinweis: Kommunikationskanäle, die größer als 3 sind, werden nur benutzt, wenn die Trajexia Tools-Software ausgeführt wird.
Argumente	–
Beispiel	**PRINT #1,"RS232"** **PRINT #2,"Port 2"**
Beispiel	**IF KEY #1 THEN GET #1,k** Überprüft die Tastatur an der RS232-Schnittstelle
Siehe auch	–

3.2.17 ABS

Typ	Mathematische Funktion
Syntax	**ABS(ausdruck)**
Beschreibung	Die Funktion **ABS** gibt den Absolutwert eines Ausdrucks zurück.
Argumente	• **ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**IF ABS(A) > 100 THEN PRINT "A liegt außerhalb des Bereichs von –100 ... 100"**
Siehe auch	–

3.2.18 ACC

Typ	Achsenbefehl
Syntax	**ACC(rate)**
Beschreibung	Stellt gleichzeitig die Beschleunigung und Verzögerung ein. Dieser Befehl bietet eine schnelle Methode, um gleichzeitig **ACCEL** und **DECEL** einzustellen. Es wird empfohlen, die Beschleunigungs- und Verzögerungsraten mit den Achsenparametern **ACCEL** und **DECEL** einzustellen.
Argumente	• **rate** Die Beschleunigungsrate in Einheiten/s^2. Sie können die Einheiten mit dem Achsenparameter **UNITS** definieren.
Beispiel	**ACC(100)** Stellt **ACCEL** und **DECEL** auf 100 Einheiten/s^2 ein.
Siehe auch	**ACCEL**, **DECEL**, **UNITS**

3.2.19 ACCEL

Typ	Achsenparameter
Syntax	**ACCEL = ausdruck**
Beschreibung	Der Achsenparameter **ACCEL** enthält die Beschleunigungsrate der Achse. Die Rate wird in Einheiten/s^2 eingestellt. Der Parameter kann jeden positiven Wert einschließlich null annehmen.
Argumente	–
Beispiel	**BASE(0)** **ACCEL = 100 ' Beschleunigungsrate einstellen** **PRINT "Beschleunigungsrate: ";ACCEL;" mm/s/s"** **ACCEL AXIS(2) = 100 ' Stellt die Beschleunigungsrate für Achse (2) ein**
Siehe auch	**ACCEL**, **DECEL**, **UNITS**

3.2.20 ACOS

Typ	Mathematische Funktion
Syntax	**ACOS(ausdruck)**
Beschreibung	Die Funktion **ACOS** gibt den Arkuskosinus des Ausdrucks zurück. Der Wert des Ausdrucks muss zwischen –1 und 1 liegen. Das Ergebnis in rad liegt zwischen 0 und Pi. Eingabewerte außerhalb des Bereichs geben 0 zurück.
Argumente	• **ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT ACOS(–1)** **3.1416**
Siehe auch	–

3.2.21 ADD_DAC

Typ Achsenbefehl

Syntax **ADD_DAC(achse)**

Beschreibung Der Befehl **ADD_DAC** kann zur dualen Regelung mit Rückführung verwendet werden, weil er ermöglicht, dass ein zweiter Drehgeber an der Servoachse eingesetzt wird. Der Befehl erlaubt es, aus dem Ausgang von zwei Servo-Regelkreisen die Summe zu bilden, um den Drehzahlsollwert für den Servotreiber zu ermitteln.
Dieser Befehl wird normalerweise in Anwendungen wie z. B. einem Walzenvorschub verwendet, wo ein zweiter Drehgeber erforderlich wäre, um den Schlupf auszugleichen.
Voraussetzung für die Verwendung von **ADD_DAC** ist, dass die zwei Achsen mit Rückführung mit einer gemeinsamen Achse verbunden sind, an der die erforderlichen Bewegungen ausgeführt werden. Normalerweise wäre dies auch möglich, indem die Bewegungen an einer der beiden Achsen mit **ADDAX** oder **CONNECT** ausgeführt werden, um eine entsprechende Sollposition (**DPOS**) für beide Achsen zu erreichen. Die Servo-Regelkreisverstärkung muss für beide Achsen eingestellt werden. Die Servo-Regelkreisausgänge werden addiert und bilden den Drehzahlsollwert-Ausgang der Servoachse. Mit **ADD_DAC(–1)** wird die Verknüpfung aufgehoben.
ADD_DAC wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.
Standard-Basisachse Hinweis:

1. Bitte beachten Sie, dass die Regelkreisverstärkung für beide Achsen sorgfältig bestimmt werden muss. Da unterschiedliche Drehgeber mit unterschiedlichen Auflösungen eingesetzt werden, sind die Verstärkungen nicht identisch.
2. Stellen Sie den Parameter **OUTLIMIT** für beide synchronisierten Achsen auf denselben Wert ein.

Argumente

- **Achse**
  Die Achse, von der die Summe des Drehzahlsollwertausgangs für die Basisachse gebildet wird. Setzen Sie das Argument auf –1, um die Verknüpfung aufzuheben und zum normalen Betrieb zurückzukehren.

Beispiel
```
BASE(0)
OUTLIMIT AXIS(1) = 15000
ADD_DAC(1) AXIS(0)
ADDAX(0) AXIS(1)
WDOG = ON
SERVO AXIS(0) = ON
SERVO AXIS(1) = ON
' Fuehrt Bewegung an Achse 0 aus
```
Dieses Beispiel zeigt die Steuerung des Servotreibers Achse 0 mit dualer Regelung mit Rückführung über Achse 0 und Achse 1.

Beispiel
```
BASE(0)
OUTLIMIT AXIS(1) = 15000
ADD_DAC(1) AXIS(0)
ADDAX(0) AXIS(1)
WDOG = ON
SERVO = OFF
S_REF = 0
BASE(1)
SERVO = ON
' Fuehrt Bewegung an Achse 1 aus
```
Dieses Beispiel zeigt die Steuerung des Servotreibers Achse 0 nur mit Drehgeber-Rückführung an Achse 1.

Siehe auch **AXIS**, **ADDAX**, **OUTLIMIT**

3.2.22 ADDAX

Typ Achsenbefehl

Syntax **ADDAX(achse)**

Beschreibung Der Befehl **ADDAX** übernimmt die Sollpositionsänderungen von der überlagernden Achse, die im Achsen-Argument spezifiziert sind und addiert sie zu jeder Bewegung, die an der Achse ausgeführt wird, an die der Befehl ausgegeben wird.
Wenn der Befehl **ADDAX** gegeben wurde, bleibt die Verknüpfung zwischen den zwei Achsen bestehen, bis sie unterbrochen wird. Mit **ADDAX(–1)** wird die Achsenverknüpfung aufgehoben. **ADDAX** ermöglicht es, dass eine Achse die Bewegungen ausführt, die für zwei zusammenaddierte Achsen spezifiziert sind. Kombinationen von mehr als zwei Achsen sind möglich, indem **ADDAX** auch auf die überlagernde Achse angewendet wird.
ADDAX wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.

Argumente
- **Achse**
  Die Achse, die als überlagernde Achse eingestellt werden soll. Setzen Sie das Argument auf –1, um die Verknüpfung aufzuheben und zum normalen Betrieb zurückzukehren.

Beispiel
```
FORWARD ' Startet eine kontinuierliche Bewegung
ADDAX(2) ' Fügt Achse 2 zur Korrektur hinzu
REPEAT
   GOSUB offset_lesen ' Anzuwendenden Offset einlesen
   MOVE(offset) AXIS(2)
UNTIL IN(2) = ON ' Bis Korrektur erfolgt ist
```
Teile werden auf ein Laufband gelegt und an anderer Stelle aufgenommen. Eine Erkennungssystem meldet, ob sich ein Teil vor oder hinter seiner Nennposition befindet und wie weit es davon entfernt ist.
In diesem Beispiel wird Achse 0 als Basisachse genommen; sie führt eine kontinuierliche Vorwärtsbewegung aus. Eine überlagernde Bewegung an Achse 2 dient dazu, Offsets gemäß dem Offset anzuwenden, der in einer Subroutine berechnet wurde.

Siehe auch **AXIS**, **OUTLIMIT**

VORSICHT
Bitte beachten Sie, dass mehrere **ADDAX** -Befehle in einem System eine gefährliche Schleife bilden können, z. B. wenn eine Achse mit einer anderen synchronisiert ist und umgekehrt. Dies kann Instabilität im System verursachen.

3.2.23 ADDAX_AXIS

Typ Achsenparameter (Nur-Lesen)

Syntax **ADDAX_AXIS**

Beschreibung Der Achsenparameter **ADDAX_AXIS** gibt die Anzahl der Achsen zurück, mit denen die Basisachse zurzeit über **ADDAX** synchronisiert ist.

Argumente –

Beispiel
```
>>BASE(0)
>> ADDAX(2)
>> PRINT ADDAX_AXIS
2.0000
```

Siehe auch **ADDAX**, **AXIS**

3.2.24 AIN

Typ Systemparameter

Syntax **AIN(analoger_kanal)**

Beschreibung +/–10 V analoge Eingangskanäle werden bereitgestellt, indem JEPMC-AN2900-Module an den MECHATROLINK-II-Bus angeschlossen werden.
Hinweis: Der Wert des Analogeingangs wird überprüft, um sicherzustellen, dass er über null liegt, auch wenn er immer positiv sein sollte. Dadurch sollen Fehlfunktionen vermieden werden, da Störungen bei den eingehenden Signalen den Wert negativ machen und einen Fehler verursachen könnten, weil eine negative Drehzahl für keine Bewegung gültig ist, außer für **FORWARD** oder **REVERSE**.

Argumente **analoger_kanal**
Analoger Eingangskanal Nummer 0,31

Beispiel
```
MOVE(–5000)
  REPEAT
    a = AIN(1)
    IF a<0 THEN a = 0
    SPEED = a * 0,25
  UNTIL MTYPE = 0
```
Die Geschwindigkeit einer Produktionslinie richtet sich nach der Geschwindigkeit, mit der das Material auf das Band geführt wird. Das Material wird über eine Lazy-Loop-Anordnung zugeführt, die mit einem Ultraschall-Höhenabtastgerät ausgestattet ist. Der Ausgang des Ultraschallsensors liegt im Bereich von 0 V bis 4 V; der Ausgang ist bei 4 V, wenn der Regelkreis am Maximum ist.

Siehe auch –

3.2.25 AND

Typ Mathematische Operation

Syntax **ausdruck1 AND ausdruck2**

Beschreibung Der Operator **AND** führt eine logische **UND**-Verknüpfung der entsprechenden Bits der Integer-Teile von zwei gültigen BASIC-Ausdrücken durch.
Die logische **UND**-Verknüpfung zwischen zwei Bits ist wie folgt definiert:
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

Argumente
- **ausdruck1**
  Jeder gültige BASIC-Ausdruck.
- **ausdruck2**
  Jeder gültige BASIC-Ausdruck.

Beispiel **VR(0) = 10 AND (2,1 * 9)**
Die Klammern werden zuerst berechnet, aber nur der Integer-Teil des Ergebnisses, 18, wird für die **AND**-Operation benutzt. Deshalb ist der Ausdruck gleich Folgendem:
VR(0) = 10 AND 18
Das **AND** ist ein Bit-Operator, daher sieht die binäre Gleichung so aus:
01010 AND 10010 = 00010
Deshalb enthält **VR(0)** den Wert 2.

Beispiel **IF MPOS AXIS(0) > 0 AND MPOS AXIS(1) > 0 THEN GOTO Zyklus1**

Siehe auch –

3.2.26 AOUT

Typ Systemparameter

Syntax **AOUT(analoger_kanal)**

Beschreibung Dieser Befehle stellt den Ausgangswert der +/–10 V analogen Ausgangskanäle ein, die durch Anschluss von JEPMC-AN2910-Modulen an den MECHATROLINK-II-Bus bereitgestellt werden. Der Einstellbereich ist [–32000, 32000] für den Spannungsbereich [–10 V, 10 V].

Argumente
- **analoger_kanal**.
  Analoger Ausgangskanal Nummer 0,31

Beispiel Kein Beispiel.

Siehe auch –

3.2.27 ASIN

Typ Mathematische Funktion

Syntax **ASIN(ausdruck)**

Beschreibung Die Funktion **ASIN** gibt den Arkussinus des Ausdrucks zurück. Das Argument muss einen Wert zwischen –1 und 1 haben. Das Ergebnis in rad liegt zwischen -Pi/2 und Pi/2. Eingabewerte außerhalb dieses Bereichs geben 0 zurück.

Argumente
- **ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel **>> PRINT ASIN(–1)**
–1.5708

Siehe auch –

3.2.28 ATAN

Typ Mathematische Funktion

Syntax **ATAN(ausdruck)**

Beschreibung Die Funktion **ATAN** gibt den Arkustangens des Arguments zurück. **Ausdruck** kann jeden beliebigen Wert annehmen. Das Ergebnis wird in rad angegeben und liegt zwischen -Pi/2 und Pi/2.

Argumente
- **ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel **>> PRINT ATAN(1)**
0.7854

Siehe auch –

3.2.29 ATAN2

Typ Mathematische Funktion

Syntax **ATAN2(ausdruck1 ,ausdruck2)**

Beschreibung Die Funktion **ATAN2** gibt den Arkustangens der komplexen Zahl ungleich null **(ausdruck2, ausdruck1)** zurück, die gleich dem Winkel zwischen einem Punkt mit den Koordinaten **(ausdruck1, ausdruck2)** und der x-Achse ist. Wenn **ausdruck2 >= 0**, ist das Ergebnis gleich dem Wert von **ATAN(ausdruck1/ausdruck2)**. Das Ergebnis wird in rad angegeben und liegt zwischen –Pi und Pi.

Argumente
- **ausdruck1**
  Jeder gültige BASIC-Ausdruck.
- **ausdruck2**
  Jeder gültige BASIC-Ausdruck.

Beispiel
```
>> PRINT ATAN2(0,1)
0.0000
```

Siehe auch –

3.2.30 ATYPE

Typ Achsenparameter

Syntax **ATYPE = Wert**

Beschreibung Der Achsenparameter **ATYPE** legt den Achsentyp für die Achse fest. Die gültigen Werte hängen von dem TJ1-Modul ab, an das der Servotreiber angeschlossen ist, der die Achse steuert. Siehe unten stehende Tabelle. Die **ATYPE**-Parameter werden vom System beim Hochfahren eingestellt. Für Achsen, die über die Servotreiber gesteuert werden, die über einen MECHATROLINK-II-Bus an das System angeschlossen sind, ist der Standardwert für ATYPE 41 (Mechatro Speed). Dies gilt für einen SigmaII Servotreiber. Standardwerte anderer Servotreiber können abweichende Standardwerte haben. Bei Achsen, die von den Servotreibern gesteuert werden, an die über den TJ1-FL02 an das System angeschlossen sind, ist der Standardwert für ATYPE 44 (Flex Achsmodul Servo).

Argumente –

Beispiel **ATYPE AXIS(1) = 45**
Mit diesem Befehl wird Achse 1 als Flex Achsmodul bzw. Drehgeber-Ausgangsachse eingestellt.

Siehe auch **AXIS**

AXIS-Typ	ATYPE-Wert	Geeignete TJ1-Baugruppe
Virtuell	0	alle
Mechatro Position	40	TJ1-ML__ (MECHATROLINK-II Master-Baugruppe)
Mechatro Drehzahl	41	TJ1-ML__
Mechatro Drehmoment	42	TJ1-ML__
Flex Achsmodul Schrittmotorausgang	43	TJ1-FL02
Flex Achsmodul Servo	44	TJ1-FL02
Flex Achsmodul Drehgeber Aus	45	TJ1-FL02
Flex Achsmodul Absolut Tamagawa	46	TJ1-FL02
Flex Achsmodul Absolut EnDat	47	TJ1-FL02
Flex Achsmodul Absolut SSI	48	TJ1-FL02

3.2.31 AUTORUN

Typ Programmbefehl

Syntax **AUTORUN**

Beschreibung Der Befehl **AUTORUN** startet alle Programme, die beim Hochfahren ausgeführt werden sollen.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **RUNTYPE**

3.2.32 AXIS

Typ Systembefehl

Syntax **AXIS(achsen_nummer)**

Beschreibung Der Modifikator **AXIS** stellt die Achse für einen einzelnen Bewegungsbefehl oder einen einzelnen Achsenparameter ein, der von einer bestimmten Achse gelesen oder zu dieser geschrieben wird. **AXIS** gilt nur für den Befehl oder die Programmzeile, in dem/der er programmiert ist. Verwenden Sie den Befehl **BASE**, um die Basisachse für alle folgenden Befehlszeilen zu ändern.

Argumente
- **achsen_nummer**
  Jeder gültige BASIC-Ausdruck, der die Achsennummer spezifiziert.

Beispiel **BASE(0)**
PRINT VP_SPEED AXIS(2)

Beispiel **MOVE(300) AXIS(0)**

Beispiel **REPDIST AXIS(1) = 100**

Siehe auch **BASE**

3.2.33 AXIS_DISPLAY

Typ Achsenparameter

Syntax **AXIS_DISPLAY = Wert**

Beschreibung Der Achsenparameter **AXIS_DISPLAY** ermöglicht die Anzeige verschiedener Daten durch die LEDs auf der Frontabdeckung des TJ1-FL02. Der Parameter bezieht sich auf zwei gelbe LEDs, die den Achsenzustand anzeigen.
Der Standardwert dieses Parameters ist beim Hochfahren für alle Achsen 0.
Die gültigen Werte sind in der unten stehenden Tabelle gezeigt.

Argumente –

Beispiel **AXIS_DISPLAY AXIS(2) = 2**
Dieser Befehl zeigt den Zustand von OUT 0 und OUT 1, die Achse 2 zugeordnet sind.

Siehe auch –

Wert von AXIS_DISPLAY	0	1	2	3
A0	REG 0	AUX IN	OUT 0	ENCODER A
A1	REG 1	ENCODER Z	OUT 1	ENCODER B
B0	REG 0	AUX IN	OUT 0	ENCODER A
B1	REG 1	ENCODER Z	OUT 1	ENCODER B

3.2.34 AXIS_ENABLE

Typ Achsenparameter

Syntax **AXIS_ENABLE = ON/OFF**

Beschreibung Der Achsenparameter **AXIS_ENABLE** wird dazu verwendet, bestimmte Achsen unabhängig von anderen Achsen zu aktivieren oder zu deaktivieren. Dieser Parameter kann für jede Achse einzeln auf EIN oder AUS gesetzt werden. Der Standardwert beim Hochfahren ist für alle Achsen EIN. Die Achsen werden aktiviert, wenn beide AXIS_ENABLE für diese Achse auf EIN gesetzt sind und **WDOG** aktiviert ist. Wird bei MECHATROLINK-II-Achsen die Einstellung für AXIS_ENABLE auf AUS gesetzt, wird der Servotreiberausgang zum Motor deaktiviert. Wird bei „Flex Achsmodul Servo Achse“ die Einstellung für AXIS_ENABLE auf AUS gesetzt, werden beide Spannungsausgänge zwangsweise auf 0 gesetzt. Bei Achsen mit der Einstellung „Flex Achsmodul Schrittmotorausgang“ und „Drehgeber Aus“ wird die Impulserzeugung an den Ausgängen blockiert, wenn AXIS_ENABLE auf AUS gesetzt ist.

Argumente –

Beispiel **AXIS_ENABLE AXIS(3) = OFF**
Dieser Befehl deaktiviert Achse 3 unabhängig von anderen Achsen im System.

Siehe auch **AXIS**, **DISABLE_GROUP**

3.2.35 AXISSTATUS

Typ Achsenparameter (Nur-Lesen)

Syntax **AXISSTATUS**

Beschreibung Der Achsenparameter **AXISSTATUS** enthält den Achsenzustand.
Die Definitionen für den Achsenparameter **AXISSTATUS** sind in der unten stehenden Tabelle gezeigt.
Der Parameter **AXISSTATUS** wird zur Behandlung von Bewegungsfehlern der Baugruppe verwendet.

Argumente –

Beispiel **IF (AXISSTATUS AND 16) > 0 THEN PRINT "In Vorwärts-Grenzwert"**

Siehe auch **AXIS**, **ERRORMASK**

Bit-nummer	Beschreibung	Wert	Zeichen (wie in Trajexia Tools verwendet)
0	–	1	–
1	Schleppfehler-Warnung	2	w
2	Kommunikationsfehler Servotreiber	4	a
3	Servotreiber-Alarm	8	m
4	Vorwärts-Grenzwert	16	f
5	Rückwärts-Grenzwert	32	r
6	Nullpunktfahrt	64	d
7	Eingabe Vorschubstopp	128	h
8	Schleppfehler-Grenzwert	256	e
9	Softwaregrenzwert Vorwärts-	512	x
10	Softwaregrenzwert Rückwärts-	1024	y
11	Bewegung abbrechen	2048	c
12	Überdrehzahl Drehgeber- Ausgang	4096	o

3.2.36 B_SPLINE

Typ Achsenbefehl

Syntax **B_SPLINE(typ, daten_ein, anzahl_ein, daten_aus, #erweiterung)**

Beschreibung Erweitert ein bestehendes Profil, das im TABLE-Speicher gespeichert ist, mit der mathematischen B-Spline-Funktion um einen konfigurierbaren Erweiterungsfaktor zu einem anderen Bereich in der TABLE.
Dieser Befehl wird idealerweise verwendet, wenn das Quellprofil **CAM** zu grob ist und in eine größere Anzahl von Punkten extrapoliert werden muss.

Argumente
- **Typ**
  Reserviert für zukünftige Erweiterung. Stets auf 1 setzen.
- **Daten_ein**
  Position in TABLE, an der das Quellprofil gespeichert ist.
- **Anzahl_ein**
  Anzahl von Punkten im Quellprofil.
- **Daten_aus**
  Position in TABLE, an der das erweiterte Profil gespeichert werden soll.
- **Erweiterungs_verhältnis**
  Das Erweiterungsverhältnis. Wenn z. B. das Quellprofil 100 Punkte umfasst und **Erweiterungs_verhältnis** auf 10 gesetzt wird, umfasst das Ergebnisprofil 1000 Punkte (100 * 10).

Beispiel Kein Beispiel.

Siehe auch –

3.2.37 BASE

Typ Achsenbefehl

Syntax **BASE**
BASE(achse_1 [,achse_2 [, achse_3 [, achse_4 [, achse_...]]]])
BA
BA(achse_1 [,achse_2 [, achse_3 [, achse_4 [, achse_...]]]])

Beschreibung Der Befehl **BASE** dient dazu, die Basisachse oder eine spezifizierte Achsenfolge-Gruppe einzustellen. Alle nachfolgenden Bewegungsbefehle und Achsenparameter wirken sich auf die Basisachse oder die spezifizierte Achsengruppe aus, wenn nicht der Befehl **AXIS** gegeben wird, um eine Achse temporär zu spezifizieren. Die Basisachse gilt, bis sie wieder mit **BASE** geändert wird.
Jeder BASIC-Prozess kann seine eigene Achsengruppe haben, und für jedes Programm kann unabhängig eine eigene Achsengruppe eingestellt werden. Verwenden Sie den Modifikator **PROC**, um auf Parameter für einen Task zuzugreifen.
Die Reihenfolge der **BASE**-Gruppierung kann festgelegt werden, indem die Abfolge der Achsen ausdrücklich zugeordnet wird. Diese Reihenfolge wird zu Interpolationszwecken bei linearen und kreisförmigen Bewegungen mit mehreren Achsen verwendet. Die Standardeinstellung für die Achsengruppe ist (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) beim Hochfahren oder wenn ein Programm in einem Task abzulaufen beginnt. Der Befehl **BASE** gibt ohne Argumente die aktuelle Reihenfolge der Achsengruppe zurück.

Hinweis: Wenn in dem Befehl **BASE** nicht alle Achsen spezifiziert sind, wird der Befehl **BASE** die verbleibenden Werte automatisch „eintragen“. Zuerst trägt er alle verbleibenden Achsen über dem letzten angegebenen Wert ein, dann alle verbleibenden Achsen in der Abfolge.
Daher ergibt sich bei **BASE(2,6,10) folgende interne Anordnung von 16 Achsen: 2,6,10,11,12,13,14,15,0,1,3,4,5,7,8,9.**

Argumente Der Befehl kann bis zu 16 Argumente annehmen.

- **Achse_i**
  Die Anzahl der als Basisachse gesetzten Achsen und alle folgenden Achsen in der Abfolge bei Bewegungen mit mehreren Achsen.

Beispiel
```
BASE(1)
UNITS = 2000 ' Stellt den Einheiten-Umwandlungsfaktor für Achse 1 ein
SPEED = 100 ' Stellt die Drehzahl für Achse 1 ein
ACCEL = 5000 ' Stellt die Beschleunigungsrate für Achse 1 ein
BASE(2)
UNITS = 2000 ' Stellt den Einheiten-Umwandlungsfaktor für Achse 2 ein
SPEED = 125 ' Stellt die Drehzahl für Achse 2 ein
ACCEL = 10000 ' Stellt die Beschleunigungsrate für Achse 2 ein
```
Es ist möglich, jede Achse mit ihrer eigenen Drehzahl, Beschleunigung und anderen Parametern zu programmieren.

Beispiel
```
BASE(0)
MOVE(100,–23,1,1250)
```
In diesem Beispiel bewegen sich die Achsen 0, 1 und 2 mit der für Achse 0 eingestellten Drehzahl und Beschleunigung zu den spezifizierten Positionen. **BASE(0)** stellt als Basisachse die Achse 0 ein, die die drei von **MOVE** verwendeten Achsen sowie die Drehzahl und Beschleunigungsrate bestimmt.

Beispiel
```
>>BASE(0,2,1)
```
In der Befehlszeile kann die Basisachsen-Reihenfolge angezeigt werden, indem **BASE** eingegeben wird.

Beispiel
```
>> RUN "PROGRAM",3
>> BASE PROC(3)(0,2,1)
```
Verwenden Sie den Modifikator **PROC**, um die Basisachsen-Reihenfolge in einem Task anzeigen zu lassen.

Beispiel
```
>>BASE(2)
>> PRINT BASE
2.0000
```
Der Druckbefehl für **BASE** gibt die aktuell ausgewählte Basisachse zurück.

Siehe auch **AXIS**

3.2.38 BASICERROR

Typ Systembefehl

Syntax **BASICERROR**

Beschreibung Der Befehl **BASICERROR** kann dazu verwendet werden, eine Routine zu starten, wenn ein Laufzeitfehler in einem Programm auftritt. **BASICERROR** kann nur als Teil eines **ON ... GOSUB** oder **ON ... GOTO**-Befehls verwendet werden. Dieser Befehl kann nur einmal in dem BASIC-Programm ausgeführt werden. Wenn mehrere Befehle verwendet werden, hat nur der zuletzt ausgeführte Gültigkeit.

Argumente –

Beispiel
```
ON BASICERROR GOTO fehler_routine
...
kein_fehler = 1
STOP
fehler_routine:
 IF kein_fehler = 0 THEN
  PRINT "Der Fehler ";RUN_ERROR[0];
  PRINT "trat auf in Zeile ";ERROR_LINE[0]
 ENDIF
 STOP
```
Wenn in diesem Beispiel ein Fehler in einem BASIC-Befehl auftritt, wird die Fehlerroutine ausgeführt.
Die Anweisung **IF** soll verhindern, dass das Programm die Fehlerroutine startet, wenn es normal beendet wird.

Siehe auch **ERROR_LINE**, **ON**, **RUN_ERROR**.

3.2.39 BATTERY_LOW

Typ Systemparameter (Nur-Lesen)

Syntax **BATTERY_LOW**

Beschreibung Dieser Parameter gibt den aktuellen Zustand der Batterie zurück. Wenn **BATTERY_LOW = ON**, muss die Batterie gewechselt werden. Wenn **BATTERY_LOW = OFF**, ist der Batteriezustand in Ordnung.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.40 BREAK_RESET

Typ Systembefehl

Syntax **BREAK_RESET "Programm_name"**

Beschreibung Wird von Trajexia Tools verwendet, um alle Abbruchpunkte aus dem spezifizierten Programm zu entfernen.

Argumente
- **programm_name**
  Der Name des Programms, aus dem Sie alle Abbruchpunkte entfernen wollen.

Beispiel **BREAK_RESET "einfacherTest"**
Entfernt alle Abbruchpunkte aus dem Programm **einfacherTest**.

Siehe auch –

3.2.41 CAM

Typ Achsenbefehl

Syntax **CAM(start_punkt, end_punkt, tabellen_multiplikator, strecke)**

Beschreibung Der Befehl **CAM** dient dazu, eine Bewegung einer Achse zu erzeugen, die einem Positionsprofil folgt, das im TABLE-Variablen-Array gespeichert ist. Die TABLE-Werte sind relativ zum Startpunkt Absolutpositionen, die in Drehgeber-Flanken spezifiziert sind. Das TABLE-Array wird mit dem Befehl **TABLE** spezifiziert.
Die Bewegung kann mit jeder beliebigen Anzahl von Punkten von 2 bis 64000 definiert werden. Der TJ1-MC__ bewegt sich ständig zwischen den Werten in der Tabelle, sodass mit einer bestimmten Anzahl von Punkten ein weiches Profil definiert werden kann. Zwei oder mehr **CAM**-Befehle können gleichzeitig ausgeführt werden, wobei sie die gleichen oder überlappende Werte im TABLE-Array verwenden. Das TABLE-Profil wird einmal abgefahren. **CAM** setzt voraus, dass das Startelement im TABLE-Array den Wert null hat. Das Abstandsargument bestimmt zusammen mit den Parametern **SPEED** und **ACCEL** die Geschwindigkeit der Bewegung durch das TABLE-Array. Bitte beachten Sie: Damit das **CAM**-Profil genau befolgt wird, muss der Parameter **ACCEL** der Achse mindestens 1000 mal größer sein als der Parameter **SPEED**.
CAM wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.

Argumente

- **start_punkt**
  Die Adresse des ersten Elements im TABLE-Array, das verwendet werden soll.
  Durch die Möglichkeit, den Startpunkt zu spezifizieren, können im TABLE-Array mehr als ein Profil und/oder andere Informationen gespeichert werden.
- **end_punkt**
  Die Adresse des letzten Elements im TABLE-Array.
- **tabellen_multiplikator**
  Der Wert für den Tabellenmultiplikator wird dazu verwendet, die in der Tabelle gespeicherten Werte zu skalieren. Da die Table-Werte in Drehgeber-Flanken spezifiziert sind, können Sie dieses Argument zum Einstellen der Werte verwenden, z. B. für den Einheiten-Umwandlungsfaktor (der durch den Parameter **UNITS** vorgegeben wird).
- **strecke**
  Ein in Benutzereinheiten angegebener Faktor, der die Geschwindigkeit der Bewegung durch die Tabelle steuert. Die zur Ausführung von **CAM** benötigte Zeit hängt von der aktuellen Achsendrehzahl und dieser Distanz ab. Nehmen wir z. B. an, das System ist in mm programmiert, die Drehzahl auf 10 mm/s eingestellt und die Beschleunigung ausreichend hoch. Wird eine Distanz von 100 mm spezifiziert, braucht **CAM** 10 Sekunden für die Ausführung.
  Der Parameter **SPEED** in der Basisachse erlaubt eine Modifizierung der Geschwindigkeit der Bewegung, wenn **CAM** verwendet wird.

Hinweis: Wird der Befehl **CAM** ausgeführt, wird der Parameter **ENDMOVE** auf das Ende der vorherigen Bewegung eingestellt.

Beispiel

Nehmen wir an, es ist eine Bewegung erforderlich, die folgender Positionsgleichung folgt: t(x) = x * 25 + 10000(1-*cos*(x)). Hier ist x in Grad angegeben. Dieses Beispiel ist für eine TABLE, mit der eine einfache Schwingung mit einer konstanten Drehzahl ausgeführt wird. Um die TABLE zu laden und den Zyklus kontinuierlich ablaufen zu lassen, kann der folgende Code verwendet werden

```
GOSUB Cam-Tabelle
schleife:
  CAM(1,19,1,200)
  GOTO schleife
```

Die Subroutine **Cam-Tabelle** lädt die Daten in der unten stehenden Tabelle in das TABLE-Array.

Siehe auch

ACCEL, **AXIS**, **CAMBOX**, **SPEED**, **TABLE**.

TABLE-Position	Grad	Wert
1	0	0
2	20	1103
3	40	3340
4	60	6500
5	80	10263
6	100	14236
7	120	18000
8	140	21160
9	160	23396
10	180	24500
11	200	24396
12	220	23160
13	240	21000
14	260	18236
15	280	15263
16	300	12500
17	320	10340
18	340	9103
19	360	9000

3.2.42 CAMBOX

Typ Achsenbefehl

Syntax **CAMBOX(start_punkt, end_punkt, tabellen_multiplikator, link_strecke, link_achse [, link_option [, link_position]])**

Beschreibung Der Befehl **CAMBOX** dient dazu, eine Bewegung einer Achse zu erzeugen, die einem Positionsprofil folgt, das im TABLE-Variablen-Array gespeichert ist. Die Bewegung ist mit der gemessenen Bewegung einer anderen Achse synchronisiert, um ein ständig variables Software-Getriebe zu bilden.
Die TABLE-Werte sind relativ zum Startpunkt Absolutpositionen, die in Drehgeber-Flanken spezifiziert sind.
Das TABLE-Array wird mit dem Befehl **TABLE** spezifiziert. Die Bewegung kann mit jeder beliebigen Anzahl von Punkten von 2 bis 64000 definiert werden. Durch die Möglichkeit, den Startpunkt zu spezifizieren, können im TABLE-Array mehr als ein Profil und/oder andere Informationen gespeichert werden. Der TJ1-MC__ bewegt sich ständig zwischen den Werten in der Tabelle, sodass mit einer bestimmten Anzahl von Punkten ein weiches Profil definiert werden kann. Zwei oder mehr **CAMBOX**-Befehle können gleichzeitig ausgeführt werden, wobei sie die gleichen oder überlappende Werte im TABLE-Array verwenden.
Der Befehl **CAMBOX** setzt voraus, dass das Startelement der TABLE den Wert null hat. Beachten Sie auch, dass der Befehl **CAMBOX** je nach Richtung der Master-Achse das Abfahren der Tabelle in Vorwärts- und Rückwärtsrichtung erlaubt.
Das Argument **link_option** kann dazu benutzt werden, verschiedene Optionen zum Starten des Befehls und einen kontinuierlichen **CAM**-Befehl zu spezifizieren. Wenn z. B. **link_option** auf 4 gesetzt wird, funktioniert **CAMBOX** wie ein „physikalischer“ **CAM**-Befehl.
CAMBOX wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.
Hinweis: Während der Ausführung von **CAMBOX** wird der Parameter **ENDMOVE** auf das Ende der vorherigen Bewegung eingestellt.
Der Achsenparameter **REMAIN** speichert die verbleibende Distanz zur Linkachse.

Argumente
- **start_punkt**
  Die Adresse des ersten Elements im TABLE-Array, das verwendet werden soll.
- **end_punkt**
  Die Adresse des letzten Elements im TABLE-Array.
- **tabellen_multiplikator**
  Der Wert für den Tabellenmultiplikator wird dazu verwendet, die in der Tabelle gespeicherten Werte zu skalieren. Da die TABLE-Werte in Drehgeber-Flanken spezifiziert sind, können Sie dieses Argument zum Einstellen der Werte verwenden, z. B. für den Einheiten-Umwandlungsfaktor (der durch den Parameter UNITS vorgegeben wird).
- **link_strecke**
  Wenn die Distanz in Benutzereinheiten angegeben ist, muss die Linkachse die spezifizierte Ausgangsbewegung vollständig ausführen. Die Linkstrecke muss als positive Distanz spezifiziert werden.
- **link_achse**
  Die Achse, mit der synchronisiert werden soll.
- **link_option**
  Siehe Tabelle unten.
- **link_position**
  Die absolute Position, bei der **CAMBOX** startet, wenn **link_option** auf 2 gesetzt ist.

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **CAM**, **REP_OPTION**, **TABLE**

Wert link_option	Beschreibung
1	Der Link startet, wenn an der Linkachse ein Druckmarkenereignis auftritt.
2	Der Link startet an einer absoluten Position auf der Linksachse (siehe **link_position**).
4	**CAMBOX** wird automatisch bidirektional wiederholt. Diese Option wird aufgehoben, indem Bit 1 des Parameters **REP_OPTION** gesetzt wird (**REP_OPTION = REP_OPTION OR 2**).
5	Kombination der Optionen 1 und 4.
6	Kombination der Optionen 2 und 4.

3.2.43 CANCEL

Typ	Achsenbefehl
Syntax	**CANCEL[(1)]** **CA[(1)]**
Beschreibung	Der Befehl **CANCEL** bricht die aktuelle Bewegung einer Achse ab. Die Drehzahlprofil-Bewegungen (**FORWARD**, **REVERSE**, **MOVE**, **MOVEABS**, **MOVECIRC**, **MHELICAL** und **MOVEMODIFY**) werden mit der Verzögerungsrate verzögert, die durch den Parameter **DECEL** vorgegeben ist und dann gestoppt. Andere Bewegungen werden sofort gestoppt. Der Befehl **CANCEL** löscht den Inhalt des aktuellen Bewegungs-Befehlsspeichers (**MTYPE**). Der Befehl **CANCEL(1)** löscht den Inhalt des Befehlsspeicher für die nächste Bewegung (**NTYPE**), ohne sich auf die aktuelle Bewegung im Befehlsspeicher **MTYPE** auszuwirken. **CANCEL** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Hinweis: • **CANCEL** bricht nur die zurzeit ausgeführte Bewegung ab. Wenn sich weitere Bewegungen im Befehlsspeicher befinden, werden sie geladen. • Während der Verzögerung der aktuellen Bewegung werden weitere **CANCEL**-Befehle ignoriert. • **CANCEL(1)** bricht die Bewegung ab, die sich im Befehlsspeicher für die nächste Bewegung befindet. Alle Bewegungen, die in den Task-Bewegungsspeichern gespeichert sind und durch die Variable **PMOVE** angezeigt werden, können in den Bewegungsspeicher geladen werden, sobald die im Bewegungsspeicher befindliche Bewegung gelöscht ist.
Argumente	–
Beispiel	**FORWARD** **WA(10000)** **CANCEL**
Beispiel	**MOVE(1000)** **MOVEABS(3000)** **CANCEL ' Bricht die Bewegung nach 3000 ab und führt stattdessen Bewegung nach 4000 aus.** **MOVEABS(4000)** Bitte beachten Sie, dass der Befehl **MOVEMODIFY** in diesem Fall eine bessere Lösung zum Modifizieren von Endpunkten von Bewegungen ist.
Siehe auch	**AXIS**, **MTYPE**, **NTYPE**, **PMOVE**, **RAPIDSTOP**

3.2.44 CHECKSUM

Typ	Systemparameter (Nur-Lesen)
Syntax	**CHECKSUM**
Beschreibung	Der Parameter **CHECKSUM** enthält die Prüfsumme für die Programme im RAM. Beim Hochfahren wird die Prüfsumme neu berechnet und mit dem zuvor gespeicherten Wert verglichen. Wenn die Prüfsumme falsch ist, startet das Programm nicht.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.45 CHR

Typ	E/A-Befehl
Syntax	**CHR(x)**
Beschreibung	Der Befehl **CHR** dient dazu, einzelne ASCII-Zeichen zu senden, die über Zahlen aufgerufen werden. **PRINT CHR(x)** entspricht **PUT(x)** in einigen anderen BASIC-Versionen.
Argumente	• **x** Ein BASIC-Ausdruck.
Beispiel	**>>PRINT CHR(65);** **A**
Siehe auch	–

3.2.46 CLEAR

Typ	Systembefehl
Syntax	**CLEAR**
Beschreibung	Der Befehl **CLEAR** setzt alle globalen VR-Variablen auf 0. Wenn er in einem Programm verwendet wird, setzt er auch die lokalen Variablen im aktuellen Task auf 0.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	• **RESET**, **VR**

3.2.47 CLEAR_BIT

Typ	Systembefehl
Syntax	**CLEAR_BIT(bit_nummer, vr_nummer)**
Beschreibung	Der Befehl **CLEAR_BIT** setzt das spezifizierte Bit in der spezifizierten VR-Variablen auf 0. Andere Bits in der Variablen behalten ihre Werte.
Argumente	• **bit_nummer** Die Nummer des Bits, das zurückgesetzt werden soll. Bereich: 0–23. • **vr_nummer** Die Nummer der VR-Variablen, für die das Bit zurückgesetzt wird. Bereich: 0–1023.
Beispiel	Kein Beispiel.
Siehe auch	**READ_BIT**, **SET_BIT**, **VR**.

3.2.48 CLEAR_PARAMS

Typ	Systembefehl
Syntax	**CLEAR_PARAMS**
Beschreibung	Setzt alle im Flash-EPROM gespeicherten Variablen und Parameter auf ihre Standardwerte zurück. CLEAR_PARAM kann nicht ausgeführt werden, wenn der Controller verriegelt ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.49 CLOSE_WIN

Typ	Achsenparameter
Syntax	**CLOSE_WIN** **CW**
Beschreibung	Der Achsenparameter **CLOSE_WIN** definiert das Ende des Fensters, in dem oder außerhalb von dem eine Druckmarke erwartet wird. Der Wert wird in Benutzereinheiten angegeben.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **OPEN_WIN**, **REGIST**, **UNITS**.

3.2.50 CLUTCH_RATE

Typ	Achsenparameter
Syntax	**CLUTCH_RATE**
Beschreibung	Der Achsenparameter **CLUTCH_RATE** definiert die Geschwindigkeit des Verbindungsaufbaus, wenn der Befehl **CONNECT** verwendet wird. Die Rate ist definiert als Betrag der Häufigkeit pro Sekunde. Standardmäßig ist der Parameter auf einen hohen Wert gesetzt (1000000), um die Kompatibilität mit früheren TJ1-MC__-Baugruppen zu gewährleisten. Hinweis: Die Operation, die **CLUTCH_RATE** verwendet, ist in der Position nicht deterministisch. Falls erforderlich, verwenden Sie stattdessen den Befehl **MOVELINK**, um unnötige Phasendifferenzen zwischen der Basisachse und synchronisierten Achsen zu vermeiden.
Argumente	–
Beispiel	**CLUTCH_RATE = 4** Diese Einstellung bewirkt, dass es eine Sekunde bis zum Erreichen der vollständigen Verbindung dauert, wenn der Befehl **CONNECT(4,1)** ausgeführt wird.
Siehe auch	**AXIS**, **CONNECT**, **MOVELINK**.

3.2.51 COMMSERROR

Typ Systemparameter (Nur-Lesen)

Syntax **COMMSERROR**

Beschreibung Der Parameter **COMMSERROR** enthält alle Kommunikationsfehler, die seit der letzten Initialisierung aufgetreten sind.
Die Bits in **COMMSERROR** sind in der nachstehenden Tabelle aufgeführt.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

Bit-	Beschreibung	Fehlerposition
8	Port 1 Rx Daten bereit	Serielle Schnittstelle 1
9	Port 1 Rx Überlauf	Serielle Schnittstelle 1
10	Port 1 Paritätsfehler	Serielle Schnittstelle 1
11	Port 1 Rx Frame-Fehler	Serielle Schnittstelle 1
12	Port 2 Rx Daten bereit	Serielle Schnittstelle 2
13	Port 2 Rx Überlauf	Serielle Schnittstelle 2
14	Port 2 Paritätsfehler	Serielle Schnittstelle 2
15	Port 2 Rx Frame-Fehler	Serielle Schnittstelle 2

3.2.52 COMMSTYPE

Typ Slot-Parameter

Syntax **COMMSTYPE SLOT(baugruppen_nummer)**

Beschreibung Dieser Parameter gibt den Baugruppentyp in einer Controller-Baugruppe zurück. In der nachstehenden Tabelle sind die zurückgegebenen Werte aufgeführt.

Argumente
- **baugruppen_nummer**
  Die Baugruppennummern sind 0 bis 6, wobei 0 die Einheit unmittelbar rechts vom TJ1-MC__ ist.

Beispiel Kein Beispiel.

Siehe auch –

Zurückgegebener Wert	Beschreibung
0	Nicht verwendete Baugruppe
31	TJ1-ML__
33	TJ1-FL02
34	TJ1-PRT
35	TJ1-DRT

3.2.53 COMPILE

Typ Programmbefehl

Syntax **COMPILE**

Beschreibung Der Befehl **COMPILE** erzwingt die Kompilierung des aktuellen Programms zu Zwischen-Code. Das Programm wird vor seiner Ausführung oder bei Auswahl eines anderen Programms automatisch von der Systemsoftware kompiliert.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.54 CONNECT

Typ Achsenbefehl

Syntax **CONNECT(verhaeltnis, antriebs_achse)**
CO(verhaeltnis, antriebs_achse)

Beschreibung Der Befehl **CONNECT** verbindet die Sollposition der Basisachse mit den gemessenen Bewegungen der Achse, die für **Antriebs_achse** spezifiziert sind, um ein elektronisches Getriebe zu erzeugen.
Das Verhältnis kann jederzeit geändert werden, indem ein weiterer **CONNECT**-Befehl für dieselbe Achse ausgeführt wird. Um die Antriebsachse zu ändern, muss der **CONNECT**-Befehl zuerst aufgehoben werden. **CONNECT** bei anderen Antriebsachsen wird ignoriert. Der Befehl **CONNECT** kann mit einem **CANCEL**- oder **RAPIDSTOP**-Befehl aufgehoben werden. Der Achsenparameter **CLUTCH_RATE** kann dazu verwendet werden, die Geschwindigkeit einzustellen mit der Achsen beim Befehl CONNECT verbunden werden.
CONNECT wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.

Argumente
- **Verhältnis**
  Das Verbindungsverhältnis des Getriebes. Das Verhältnis wird spezifiziert als Drehgeber-Flankenverhältnis (keine Einheiten). Es enthält die Anzahl der Encoderflanken, die die Basisachse benötigt, um sich inkremental im Verhältnis zur Antriebsachse zu bewegen. Der Verhältniswert kann positiv oder negativ sein und hat eine 16-Bit-Bruchauflösung.
- **Antriebs_achse**
  Die Master-Achse, die die Basisachse antreibt.

Beispiel Bei einer Durchlaufpresse muss sich eine Walze mit einer Geschwindigkeit drehen, die einem Viertel des von einem Drehgeber gemessenen Wertes entspricht, der an dem Zufuhr-Förderband montiert ist. Die Walze ist mit Achse 0 verdrahtet. Ein Eingangskanal überwacht die Drehgeber-Impulse vom Förderband und bildet Achse 1. Der folgende Code kann verwendet werden:

```
BASE(1)
SERVO = OFF ' Diese Achse dient zur Ueberwachung des
Foerderbandes
BASE(0)
SERVO = ON
CONNECT(0,25,1)
```

Siehe auch **AXIS**, **CANCEL**, **CLUTCH_RATE**, **CONNECT**, **RAPIDSTOP**.

3.2.55 CONSTANT

Typ Systembefehl

Syntax **CONSTANT "Name", Wert**

Beschreibung Definiert den Namen als Konstante, die sowohl innerhalb des Programms, das die Definition für **CONSTANT** enthält als auch von allen anderen Programmen im Trajexia Tools-Projekt verwendet werden kann. Hinweis: Das Programm mit der Definition für **CONSTANT** muss laufen, bevor der Name in anderen Programmen verwendet wird. Wenn die CONSTANT Anweisung ausgeführt wird darf nur das Programm mit der CONSTANT Anweisung laufen; andernfalls wird eine Fehlermeldung angezeigt, und das Programm stoppt, wenn versucht wird, diesen Befehl auszuführen. Für einen schnellen Start sollte dieses Programm außerdem der einzige Prozess sein, der beim Hochfahren läuft.
Wenn CONSTANT definiert wird, bleibt die Definition bis zum nächsten Zurücksetzen des TJ1-MC__ aktiv; dies kann erfolgen, indem die Versorgungsspannung aus und wieder eingeschaltet wird oder indem der EX-Befehl ausgeführt wird.
Es können maximal 128 **CONSTANT**-Befehle definiert werden.

Argumente
- **Name**
  Jeder benutzerdefinierte Name, der alphanumerische oder Unterstrich-Zeichen enthält.
- **Wert**
  Der Wert, der **Name** zugeordnet ist.

Beispiel

```
CONSTANT "nak",$15
CONSTANT "start_schaltflaeche",5
IF IN(start_schaltflaeche) = ON THEN OP(led1,ON)
IF key_char = nak THEN GOSUB no_ack_received
```

Siehe auch –

3.2.56 CONTROL

Typ Systemparameter (Nur-Lesen)

Syntax **CONTROL**

Beschreibung Der Parameter **CONTROL** enthält den an das System angeschlossenen Typ des TJ1-MC__. Der Wert dieses Systemparameters für den TJ1-MC__ ist 262.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.57 COPY

Typ Programmbefehl

Syntax **COPY programm_name neuer_programm_name**

Beschreibung Der Befehl **COPY** kopiert ein im Controller bestehendes Programm zu einem neuen Programm mit dem spezifizierten Namen. Der Programmname kann ohne Anführungszeichen spezifiziert werden.
Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann der Benutzer den Befehl aus dem Menü **Programm** auswählen.

Argumente
- **programm_name**
  Name des zu kopierenden Programms.
- **neuer_programm_name**
  Name des neuen Programms.

Beispiel
```
>> COPY "prog" "neuesprog"
```

Siehe auch **DEL**, **NEW**, **RENAME**.

3.2.58 COS

Typ Mathematische Funktion

Syntax **COS(ausdruck)**

Beschreibung Die Funktion **COS** gibt den Kosinus des Ausdrucks zurück. Die Eingabewerte sind in rad und können jeden beliebigen Wert annehmen. Der Ergebniswert liegt im Bereich zwischen –1 und 1.

Argumente
- **Ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel
```
>> PRINT COS(0)
1.0000
```

Siehe auch –

3.2.59 CREEP

Typ Achsenparameter

Syntax **CREEP**

Beschreibung Der Achsenparameter **CREEP** enthält die Schleichganggeschwindigkeit für die Achse. Die Schleichganggeschwindigkeit wird für den langsamen Teil einer Nullpunkt-Suchsequenz verwendet. **CREEP** kann jeden positiven Wert einschließlich null annehmen.
Die Schleichganggeschwindigkeit wird in Einheiten mit dem Einheiten-Umwandlungsfaktor **UNITS** eingegeben. Wenn z. B. der Einheiten-Umwandlungsfaktor auf die Anzahl der Drehgeber-Flanken/Zoll eingestellt ist, wird die Drehzahl in Zoll eingestellt.

Argumente –

Beispiel
```
BASE(2)
CREEP = 10
SPEED = 500
DATUM(4)
CREEP AXIS(1) = 10
SPEED AXIS(1) = 500
DATUM(4) AXIS(1)
```

Siehe auch **AXIS**, **DATUM**, **UNITS**.

3.2.60 D_GAIN

Typ Achsenparameter

Syntax **D_GAIN**

Beschreibung Der Achsenparameter **D_GAIN** enthält den D-Anteil für die Achse.
Der D-Anteil wird berechnet, indem die Änderung im Schleppfehler mit **D_GAIN** multipliziert wird. Der Standardwert ist 0.
Fügen Sie den D-Anteil einem System hinzu, um eine sanftere Reaktion und die Verwendung einer höheren Proportionalverstärkung zu erreichen. Hohe Werte können zu Schwingungen führen.
Hinweis: Die Servo-Verstärkung darf nur geändert werden, wenn **SERVO** deaktiviert ist.

Argumente –

Beispiel	Kein Beispiel.
Siehe auch	• **AXIS**, **I_GAIN**, **OV_GAIN**, **P_GAIN**, **VFF_GAIN**.

3.2.61 D_ZONE_MAX

Typ	Systemparameter
Syntax	**D_ZONE_MAX = Wert**
Beschreibung	In Verbindung mit **D_ZONE_MIN** hält dieser Parameter den DAC-Ausgang auf Null, wenn die gewünschte Bewegung ausgeführt und die Größe des Schleppfehlers kleiner ist als der Wert für **D_ZONE_MIN**. Der Servo-Regelkreis wird reaktiviert, wenn entweder der Schleppfehler über den Wert von **D_ZONE_MAX** ansteigt oder eine neue Bewegung gestartet wird.
Argumente	–
Beispiel	**D_ZONE_MIN = 3** **D_ZONE_MAX = 10** Wenn diese beiden Parameter wie oben eingestellt sind, wird der DAC-Ausgang auf Null gehalten, wenn die Bewegung abgeschlossen ist und der Schleppfehler unter den Wert 3 abfällt. Wenn eine Bewegung neu gestartet wird oder wenn der Schleppfehler über den Wert 10 ansteigt, wird der Servo-Regelkreis reaktiviert.
Siehe auch	**D_ZONE_MIN**.

3.2.62 D_ZONE_MIN

Typ	Systemparameter
Syntax	**D_ZONE_MIN = Wert**
Beschreibung	In Verbindung mit **D_ZONE_MAX** hält dieser Parameter den DAC-Ausgang auf Null, wenn die gewünschte Bewegung ausgeführt und die Größe des Schleppfehlers kleiner ist als der Wert für D_ZONE_MIN. Der Servo-Regelkreis wird reaktiviert, wenn entweder der Schleppfehler über den Wert von **D_ZONE_MAX** ansteigt oder eine neue Bewegung gestartet wird.
Argumente	–
Beispiel	**D_ZONE_MIN = 3** **D_ZONE_MAX = 10** Wenn diese beiden Parameter wie oben eingestellt sind, wird der DAC-Ausgang auf Null gehalten, wenn die Bewegung abgeschlossen ist und der Schleppfehler unter den Wert 3 abfällt. Wenn eine Bewegung neu gestartet wird oder wenn der Schleppfehler über den Wert 10 ansteigt, wird der Servo-Regelkreis reaktiviert.
Siehe auch	**D_ZONE_MAX**.

3.2.63 DAC

Siehe **S_REF**.

3.2.64 DAC_OUT

Siehe **S_REF_OUT**.

3.2.65 DAC_SCALE

Typ	Achsenparameter
Syntax	**DAC_SCALE**
Beschreibung	Der Parameter hat zwei Aufgaben: 1. Er wird beim Hochfahren der integrierten Achsen des Systems auf den Wert 16 gesetzt. Dadurch werden die für die höhere Auflösung von **DAC** geltenden Werte skaliert, so dass die an der Achse erforderlichen Verstärkungen ähnlich wie die sind, die bei den anderen Controllern erforderlich sind. 2. **DAC_SCALE** kann negativ gesetzt werden (–16), um die Polarität des DAC-Ausgangsignals umzukehren. Ist der Servo deaktiviert, ist die Größe von **DAC_SCALE** unwichtig, da die angelegte Spannung durch den Parameter **DAC** geregelt wird. Die Polarität wird jedoch durch **DAC_SCALE** umgekehrt.
Argumente	–
Beispiel	**DAC_SCALE AXIS(3) = –16**
Siehe auch	**DAC**, **S_REF**.

3.2.66 DATE

Typ	Systemparameter
Syntax	**DATE**
Beschreibung	Gibt das aktuelle Datum der Trajexia-Echtzeituhr zurück oder stellt dieses ein. Die Zahlen können im Format TT:MM:JJ oder TT:MM:JJJJ eingegeben werden.
Argumente	–
Beispiel	**DATE = 20:10:05** oder **DATE = 20:10:2005**
Beispiel	**>>PRINT DATE** **36956** Dadurch wird die Zahlenangabe für den aktuellen Tag ausgedruckt. Diese Zahl ist die Anzahl der Tage seit dem 1. Januar 1900, wobei der 1. Januar 1900 der Zahl 1 entspricht.
Siehe auch	–

3.2.67 DATE$

Typ	Systembefehl
Syntax	**DATE$**
Beschreibung	Druckt das aktuelle Datum im Format TT/MM/JJ als Zeichenkette aus. Das Jahr wird zweistellig angegeben.
Argumente	–
Beispiel	**PRINT #1,DATE$** Dadurch wird das Datum im folgenden Format ausgedruckt: 20/10/05
Siehe auch	–

3.2.68 DATUM

Typ	Achsenbefehl
Syntax	**DATUM(sequenz)**
Beschreibung	Der Befehl **DATUM** führt eine von 6 Nullpunkt-Suchsequenzen durch, um eine Achse auf eine Absolutposition zu setzen und um den Schleppfehler zu löschen: **DATUM** verwendet sowohl die Schleichganggeschwindigkeit als auch die gewünschte Geschwindigkeit für die Nullpunktsuche. Die Schleichganggeschwindigkeit wird mit dem Achsenparameter **CREEP** und die gewünschte Geschwindigkeit mit dem Achsenparameter **SPEED** eingestellt. Die Eingangsnummer für den Nullpunktschalter, die für die Sequenzen 3 bis 6 verwendet wird, wird durch den Parameter **DATUM_IN** eingestellt. **DATUM** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Hinweis: Der mit dem Parameter **DATUM_IN** eingestellte Nullpunkteingang ist in aktiviertem Zustand null, d. h. der Nullpunktschalter wird betätigt, wenn der Eingang auf AUS steht. Die Endschaltereingänge für Feedhold, Rückwärts-JOG, Vorwärts-JOG, Vorwärts und Rückwärts sind ebenfalls in aktiviertem Zustand null Wenn Eingänge in aktiviertem Zustand null sind, bedeutet das gleichzeitig eine drahtbuchsichere Verdrahtung.
Argumente	• **Folge** Siehe Tabelle unten.
Beispiel	Kein Beispiel.
Siehe auch	**ACCEL**, **AXIS**, **AXISSTATUS**, **CREEP**, **DATUM_IN**, **DECEL**, **MOTION_ERROR**, **SPEED**.

Folgewert	Beschreibung
0	Der Befehl **DATUM(0)** löscht den Bewegungsfehler. Die aktuell gemessene Position wird als Sollposition eingestellt (dies ist besonders nützlich bei Schrittmotor-Achsen mit Positionskontrolle). DATUM(0) löscht auch für ALLE Achsen den Schleppfehler, der die Bedingung FE_LIMIT im Register **AXISSTATUS** überschritten hat. Der Befehl setzt folgende Bits in AXXISSTATUS auf null: Bit 1: Warnung Schleppfehler Bit 2: Kommunikationsfehler dezentraler Treiber Bit 3: Fehler dezentraler Treiber Bit 8: Grenzwert für Schleppfehler überschritten Bit 11: Bewegung wird abgebrochen. Bitte beachten Sie, dass der Zustand nicht gelöscht werden kann, wenn die Ursache des Problems weiterhin besteht.
1	Die Achse bewegt sich mit Schleichganggeschwindigkeit vorwärts, bis die Z-Marke erreicht ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.
2	Die Achse bewegt sich mit Schleichganggeschwindigkeit rückwärts, bis die Z-Marke erreicht ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.
3	Die Achse bewegt sich mit der gewünschten Geschwindigkeit vorwärts, bis der Nullpunktschalter erreicht ist. Die Achse bewegt sich anschließend mit Schleichganggeschwindigkeit rückwärts, bis der Nullpunktschalter zurückgesetzt ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.
4	Die Achse bewegt sich mit der gewünschten Geschwindigkeit rückwärts, bis der Nullpunktschalter erreicht ist. Die Achse bewegt sich anschließend mit Schleichganggeschwindigkeit vorwärts, bis der Nullpunktschalter zurückgesetzt ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.
5	Die Achse bewegt sich mit der gewünschten Geschwindigkeit vorwärts, bis der Nullpunktschalter erreicht ist. Die Achse bewegt sich anschließend mit Schleichganggeschwindigkeit rückwärts, bis der Nullpunktschalter zurückgesetzt ist. Die Achse bewegt sich mit Schleichganggeschwindigkeit weiter, bis die Z-Marke des Drehgebers erreicht ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.
6	Die Achse bewegt sich mit der gewünschten Geschwindigkeit rückwärts, bis der Nullpunktschalter erreicht ist. Die Achse bewegt sich anschließend mit Schleichganggeschwindigkeit vorwärts, bis der Nullpunktschalter zurückgesetzt ist. Die Achse bewegt sich mit Schleichganggeschwindigkeit weiter, bis die Z-Marke des Drehgebers erreicht ist. Die Sollposition wird dann auf 0 zurückgesetzt, und die gemessene Position wird korrigiert, um den Schleppfehler aufrechtzuerhalten.

3.2.69 DATUM_IN

Typ: Achsenparameter

Syntax: **DATUM_IN**
DAT_IN

Beschreibung: Der Achsenparameter **DATUM_IN** enthält die Eingangsnummer, die als Nullpunktschalter für den Befehl **DATUM** verwendet werden soll. Der gültige Eingangsbereich ist durch die Werte 0 bis 31 gegeben. Die Werte 0 bis 15 stellen physikalisch die Eingänge des TJ1-MC__ E/A-Steckverbinders dar und werden für alle Achsen gemeinsam benutzt. Die Werte 16 bis 27 stellen Software-Eingänge dar, die beliebig in Programmen und Befehlen verwendet werden können, z. B. IN und OP. Sie gelten ebenfalls für alle Achsen.
Die Werte 28 bis 31 werden direkt den am CN1-Steckverbinder vorhandenen Treiber-Eingängen zugewiesen und sind für jede Achse eindeutig. Welche Treiber-Eingänge den Eingängen 28 bis 31 zugewiesen werden, hängt von der Servotreiber-Einstellung in Parameter Pn81E ab. Die empfohlene Einstellung ist Pn81E = 0 x 4321, mit folgender Zuordnung:
Hinweis: Der Nullpunkteingang ist in aktiviertem Zustand null, d. h. der Nullpunktschalter wird betätigt, wenn der Eingang auf AUS steht.
Die Endschaltereingänge für Feedhold, Rückwärts-JOG, Vorwärts-JOG, Vorwärts und Rückwärts sind ebenfalls in aktiviertem Zustand null. Wenn Eingänge in aktiviertem Zustand null sind, bedeutet das gleichzeitig eine drahtbuchsichere Verdrahtung.

Sigma II:
- Eingang 28: CN1-40
- Eingang 29: CN1-41
- Eingang 30: CN1-42
- Eingang 31: CN1-43

Sigma III	• Eingang 28: CN1-13 • Eingang 29: CN1-7 • Eingang 30: CN1-8 • Eingang 31: CN1-9
Junma	• Eingang 26: CN1-2 • Eingang 27: CN1-1
	Weitere Informationen zum Einstellen des Parameters Pn81E finden Sie im Servotreiber-Handbuch. Standardmäßig ist der Parameter auf –1 eingestellt, und es sind keine Eingänge ausgewählt.
Argumente	–
Beispiel	**DATUM_IN AXIS(0) = 5**
Siehe auch	**AXIS**, **DATUM**.

3.2.70 DAY

Typ	Systemparameter
Syntax	**DAY**
Beschreibung	Gibt den aktuellen Tag als Zahl von 0 bis 6 zurück, Sonntag ist 0. **DAY** kann durch Zuordnung eingestellt werden.
Argumente	–
Beispiel	**>>DAY = 3** **>>? DAY** **3.0000**
Siehe auch	–

3.2.71 DAY$

Typ	Systembefehl
Syntax	**DAY$**
Beschreibung	Druckt den aktuellen Tag als Zeichenkette.
Argumente	–
Beispiel	**>>DAY = 3** **>>? DAY$** **Mittwoch**
Siehe auch	–

3.2.72 DECEL

Typ	Achsenparameter
Syntax	**DECEL**
Beschreibung	Der Achsenparameter **DECEL** enthält die Verzögerungsrate der Achse. Die Rate wird in Einheiten/s^2 eingestellt. Der Parameter kann jeden positiven Wert einschließlich 0 annehmen.
Argumente	–
Beispiel	**DECEL = 100 ' Verzögerungsrate einstellen** **PRINT "Verzögerungsrate ist ";DECEL;" mm/s/s"**
Siehe auch	**ACCEL**, **AXIS**, **UNITS**.

3.2.73 DEFPOS

Typ	Achsenbefehl
Syntax	**DEFPOS(pos_1 [, pos_2 [, pos_3 [, pos_4 [, ...]]]])** **DP(pos_1 [, pos_2 [, pos_3 [, pos_4 [, ...]]]])**
Beschreibung	Der Befehl **DEFPOS** definiert die aktuelle Sollposition (**DPOS**) als neue Absolutposition. Die gemessene Position (**MPOS**) wird entsprechend geändert, um den Schleppfehler aufrechtzuerhalten. **DEFPOS** wird normalerweise nach einer Nullpunkt-Suchsequenz verwendet (siehe DATUM-Befehl), da hierbei die aktuelle Position auf 0 gesetzt wird. **DEFPOS** kann jederzeit verwendet werden. Alternativ dazu kann auch der Achsenparameter **OFFPOS** benutzt werden. Dieser Parameter kann dazu verwendet werden, eine relative Anpassung der aktuellen Position vorzunehmen. **DEFPOS** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Hinweis: Die an der Achsenposition mit dem Befehl DEFPOS oder OFFPOS vorgenommenen Änderungen werden beim nächsten Servo-Update ausgeführt. Dies kann möglicherweise Probleme verursachen, wenn im selben Servozyklus, in dem DEFPOS oder OFFPOS ausgeführt werden, auch eine Bewegung initiiert wird. Das folgende Beispiel zeigt, wie der Parameter **OFFPOS** verwendet werden kann, ohne dass dieses Problem auftritt. DEFPOS-Befehle werden intern in **OFFPOS**-Positions-Offsets umgewandelt, wodurch das Problem leicht vermieden werden kann, indem wie folgt programmiert wird: **DEFPOS(100): WAIT UNTIL OFFPOS = 0: MOVEABS(0)**

Argumente Der Befehl kann bis zu 16 Argumente annehmen.
- **pos_i**
  Die Absolutposition für die (basis + i) -Achse in Benutzereinheiten. Siehe **BASE**-Befehl für die Gruppierung der Achsen.

Beispiel
```
BASE(2)
DATUM(5)
BASE(1)
DATUM(4)
WAIT IDLE
DEFPOS(–1000,–3500)
```
Die letzte Zeile definiert die aktuelle Position zu (–1000, –3500) in Benutzereinheiten. Die aktuelle Position wäre mit den beiden **DATUM**-Befehlen auf (0,0) zurückgesetzt worden.

Siehe auch **AXIS**, **DATUM**, **DPOS**, **OFFPOS**, **MPOS**, **UNITS**.

3.2.74 DEL

Typ Programmbefehl

Syntax **DEL [Programm_name]**
RM [Programm_name]

Beschreibung Der Befehl **DEL** löscht ein Programm aus dem Controller. **DEL** ohne Programmname kann dazu verwendet werden, das aktuell ausgewählte Programm zu löschen (mit **SELECT**). Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. **DEL ALL** löscht alle Programme. **DEL** kann auch dazu verwendet werden, die Tabelle zu löschen: **DEL "TABLE"**. Die Bezeichnung **"TABLE"** muss in Anführungszeichen gesetzt sein.
Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann der Benutzer den Befehl aus dem Menü **Programm** auswählen.

Argumente
- **Programm_name**
  Name des zu löschenden Programms.

Beispiel **>> DEL altesProg**

Siehe auch **COPY**, **NEW**, **RENAME**, **SELECT**, **TABLE**.

3.2.75 DEMAND_EDGES

Typ Achsenparameter (Nur-Lesen)

Syntax **DEMAND_EDGES**

Beschreibung Der Achsenparameter **DEMAND_EDGES** enthält den aktuellen Wert des Achsenparameters **DPOS** in Drehgeber-Flankeneinheiten.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **DPOS**.

3.2.76 DEVICENET

Typ Systembefehl

Syntax **DEVICENET(baugruppen_nummer, 2, 1, VR_start_ausgaenge, anzahl_ausgaenge, VR_start_einaenge, anzahl_eingaenge)**
DEVICENET((baugruppen_nummer, 4, 0)

Beschreibung Die Funktion **DEVICENET** konfiguriert den TJ1-DRT für den Datenaustausch mit der DeviceNet-Master-Baugruppe und definierte Bereiche im VR-Speicher, wo der E/A-Austausch stattfindet. Die Funktion **DEVICENET** 4 gibt den Datenaustauschstatus des TJ1-DRT zurück. Eine Beschreibung der Bits im Datenaustausch-Statuswort finden Sie in der Tabelle.

Argumente
- **baugruppen_nummer**
  Spezifiziert die Baugruppennummer des TJ1-DRT im Trajexia-System.
- **VR_start_ausgaenge**
  Die Startadresse im VR-Speicher des Controllers, an der sich die Ausgangsdaten vom DeviceNet-Master befinden.
- **anzahl_ausgaenge**
  Die Anzahl der Ausgangsworte vom DeviceNet-Master im VR-Speicher.
- **VR_start_eingaenge**
  Die Startadresse im VR-Speicher des Controllers, an der sich die Eingangsdaten für den DeviceNet-Master befinden.
- **anzahl_eingaenge**
  Die Anzahl der Eingangsworte zum DeviceNet-Master im VR-Speicher.

Beispiel **DEVICENET (0,2,1,10,16,150,31)**
In diesem Beispiel ist der TJ1-DRT für den Datenaustausch mit dem DeviceNet-Master konfiguriert, und zwar mit 16 Ausgangsworten (die vom Master empfangen werden) an den Positionen VR(10) bis VR(25) und 31 Eingangsworten (die an den Master gesendet werden) an den Positionen VR(150) bis VR(180).

Siehe auch –

Bit	Wert	Beschreibung
0	0	**DEVICENET (baugruppen_nummer, 2, ...)** noch nicht ausgeführt
	1	**DEVICENET (baugruppen_nummer, 2, ...)** ohne Fehler ausgeführt
1	0	Keine DeviceNet-E/A-Verbindung
	1	DeviceNet-E/A-Verbindung läuft
2	0	VR-Variablen im Ausgangsdatenbereich sind aktualisiert worden
	1	VR-Variablen im Ausgangsdatenbereich sind noch nicht aktualisiert worden
3	0	Die DeviceNet-E/A-Verbindungsgröße stimmt mit dem Befehl **DEVICENET (baugruppen_nummer, 2,…)** überein
	1	Die DeviceNet-E/A-Verbindungsgröße stimmt nicht mit dem Befehl **DEVICENET (baugruppen_nummer, 2,…)** überein
4–7	0	Immer null
8	0	Netzwerk-Spannungsversorgung OK
	1	Netzwerk-Spannungsversorgung ausgefallen
9	0	Kein BUSOFF aufgetreten
	1	BUSOFF aufgetreten
10	0	Kein Teilnehmeradressen-Duplizierungsfehler
	1	Teilnehmeradressen-Duplizierungsfehler

3.2.77 DIR

Typ Programmbefehl

Syntax **DIR**
LS

Beschreibung Der Befehl **DIR** zeigt eine Liste von im Controller gespeicherten Programmen, die Speichergröße und den **RUNTYPE** an. DIR zeigt außerdem die verfügbare Speichergröße, die Betriebsart beim Einschalten und das aktuell ausgewählte Programm des Controllers an.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **FREE**, **POWER_UP**, **PROCESS**, **RUNTYPE**, **SELECT**.

3.2.78 DISABLE_GROUP

Typ Achsenbefehl

Syntax **DISABLE_GROUP(–1)**
DISABLE_GROUP(achse_1 [, achse_2 [, ...]])

Beschreibung Wird verwendet, um mehrere Achsen zur Fehler-Deaktivierung zusammenzugruppieren. Wenn in einer Achsengruppe ein Fehler bei **einer** auftritt, wird bei allen **AXIS_ENABLE** deaktiviert und **SERVO** deaktiviert. Es können mehrere Gruppen erstellt werden, jedoch kann eine Achse nicht zu mehrere Gruppen gehören. Alle Gruppierungen können mit **DISABLE_GROUP(–1)** gelöscht werden.

Argumente
- **Achse_i**
  Ein BASIC-Ausdruck, der die Achsennummer spezifiziert.

Beispiel
```
DISABLE_GROUP(–1)
DISABLE_GROUP(0,1,2,3)
DISABLE_GROUP(4,5,6,7)
WDOG = ON
STOP
enable_b:
FOR ax = 4 TO 7
  AXIS_ENABLE AXIS(ax) = ON
NEXT ax
```
Ein System von 8 Achsen, bei dem die Achsen 4..7 weiterlaufen, wenn bei den Achsen 0..3 ein Fehler auftritt und umgekehrt. Die Achsen können mit **DISABLE_GROUP** zusammengruppiert werden.
Hinweis: Nur zur Verwendung bei MECHATROLINK-II.

Siehe auch –

3.2.79 DISPLAY

Typ Systemparameter

Syntax **DISPLAY = Wert**

Beschreibung Legt fest, welche E/A-Kanäle durch die LEDs auf der Vorderseite angezeigt werden. Der Parameter **DISPLAY** kann dazu verwendet werden auszuwählen, welche E/A-Bank angezeigt werden soll.
Die Werte sind in der nachstehenden Tabelle aufgeführt.

Argumente –

Beispiel **DISPLAY = 5**
Zeigt die Ausgänge 8–15.

Siehe auch –

Wert	Beschreibung
0	Eingänge 0 bis 7 (Standardeinstellung)
1	Eingänge 8 bis 15
2	Eingänge 16 bis 23
3	Eingänge 24 bis 31

Wert	Beschreibung
4	Ausgänge 0 bis 7 (bei Trajexia nicht verwendet)
5	Ausgänge 8 bis 15
6	Ausgänge 16 bis 23
7	Ausgänge 24 bis 31

3.2.80 DPOS

Typ Achsenparameter (Nur-Lesen)

Syntax **DPOS**

Beschreibung Der Achsenparameter **DPOS** enthält die Sollposition in Benutzereinheiten, die durch die Bewegungsbefehle in der Lageregelung generiert wird.
Wenn der Lageregelkreis im Controller geöffnet ist (**SERVO = OFF**), wird die gemessene Position (**MPOS**) zu dem **DPOS** kopiert, um einen Schleppfehler von 0 aufrechtzuerhalten.
Der Bereich der gewünschten Position wird mit den Achsenparametern **REP_DIST** und **REP_OPTION** kontrolliert. Mit dem Befehl **DEFPOS** oder dem Achsenparameter **OFFPOS** kann der Wert eingestellt werden, ohne eine Bewegung auszuführen. **DPOS** wird beim Hochfahren auf 0 zurückgesetzt.

Argumente –

Beispiel
```
>> PRINT DPOS AXIS(0)
34.0000
```
Die obige Zeile gibt die Sollposition in Benutzereinheiten zurück.

Siehe auch **AXIS**, **DPOS**, **DEFPOS**, **DEMAND_EDGES**, **FE**, **MPOS**, **REP_DIST**, **REP_OPTION**, **OFFPOS**, **UNITS**.

3.2.81 DRIVE_ALARM

Typ Achsenbefehl

Syntax **DRIVE_ALARM(VR)**

Beschreibung Die Funktion DRIVE_ALARM liest den aktuellen Alarm des Servotreibers aus, der über MECHATROLINK-II an das Trajexia-System angeschlossen ist. Nach erfolgreicher Ausführung gibt der Befehl –1 zurück und speichert den Wert an dem Speicheradresse im VR-Speicher, der durch den VR-Parameter spezifiziert ist. Wenn der Befehl nicht ausgeführt werden kann, wird der Wert 0 zurückgegeben. Der Befehl wird im Treiber für die durch BASE eingestellte Basisachse durchgeführt. Die Basisachse kann mit dem Modifikator AXIS sowie mit allen anderen Achsenbefehlen und -parametern geändert werden. Dieser Befehl wartet auf die Antwort von der Achse; die Ausführung des Befehls kann langsam sein und die Dauer variieren. Wenn Sie eine schnelle Reaktion benötigen, verwenden Sie diesen Befehl nicht.

Argumente
- **VR**
  Bei erfolgreicher Ausführung wird der Alarmwert an der VR-Adresse gespeichert.

Beispiel
```
IF NOT DRIVE_ALARM(10) AXIS(2) THEN
PRINT "Alarm für Servotreiber kann nicht gelesen werden"
ELSE
 IF VR(10) = 0 THEN
  PRINT "Servotreiber in Ordnung"
 ELSE
  PRINT "Servo-Alarmcode: "; VR(10)
 ENDIF
ENDIF
```
In diesem Beispiel wird ein Alarm des Servotreibers von Achse 2 ausgelesen; diese Informationen werden dem Benutzer angezeigt.

Siehe auch –

3.2.82 DRIVE_CLEAR

Typ	Achsenbefehl
Syntax	**DRIVE_CLEAR**
Beschreibung	Der Befehl **DRIVE_CLEAR** löscht den Alarmzustand des Servotreibers, der über MECHATROLINK-II an das System angeschlossen ist. Mit diesem Befehl können nicht alle möglichen Alarmzustände gelöscht werden. Bestimmte Alarme können nur durch Aus- und Einschalten der Spannungsversorgung (sowohl der des TJ1-MC__ als auch der des Servotreibers) zurückgesetzt werden.)
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**DRIVE_STATUS**.

Achtung
Stellen Sie sicher, dass keine Bedienkonsole oder PC-Software an den Servotreiber angeschlossen ist, wenn dieser Befehl ausgeführt wird. Andernfalls wird die Programmausführungung unterbrochen, bis die Verbindung des anderen Gerätes mit dem Servotreiber getrennt wird.

3.2.83 DRIVE_CONTROL

Typ	Achsenparameter
Syntax	**DRIVE_CONTROL**
Beschreibung	Wird der Befehl auf eine Achse angewendet, die mit einem an den MECHATROLINK-II-Bus angeschlossenen Servotreiber verbunden ist, wählt dieser Parameter die Daten aus, die nach der unten stehenden Tabelle durch **DRIVE_MONITOR** überwacht werden sollen. Wenn eine Achse mit einen Servotreiber über den TJ1-FL02 verbunden ist, stellt dieser Parameter die Ausgänge des TJ1-FL02 ein. Stellen Sie Bit 8 dieses Parameters so ein, dass OUT 0 für eine Achse eingeschaltet ist. Stellen Sie Bit 9 dieses Parameters so ein, dass OUT 1 für eine Achse eingeschaltet ist. Bitte beachten Sie, dass die gleichen Ausgänge von dem Befehl **HW_PSWITCH** benutzt werden.
Argumente	–
Beispiel	**DRIVE_CONTROL AXIS(2) = 256** In diesem Beispiel wird OUT 0 für Achse 2 eingeschaltet, die über den TJ1-FL02 angeschlossen ist.
Siehe auch	–

Code	Beschreibung
2	Schleppfehler (dies ist der tatsächliche FE, wenn ATYPE = 40 verwendet wird)
8	Istdrehzahl (bei Atype = 41 Einheiten = Max Drehzahl/40000000H, bei anderen Atype-Baugruppen = Sollwerteinheiten/s)
9	Solldrehzahl (die gleichen Einheiten wie bei Istdrehzahl)
10	Zieldrehzahl (die gleichen Einheiten wie bei Istdrehzahl)
11	Drehmoment- (Kraft-) Sollwert (bei Atype = 42 Einheiten = Max Drehmoment/ 40000000H, bei anderen Atype-Baugruppen = % über Nenndrehmoment)
14	Mit Pn813.0 ausgewählter Monitor. Nützlich, um die Servomonitore zu überwachen (Unxxx)
15	Mit Pn813.1 ausgewählter Monitor. Nützlich, um die Servomonitore zu überwachen (Unxxx)

3.2.84 DRIVE_INPUTS

Typ	Achsenparameter
Syntax	**DRIVE_INPUTS**
Beschreibung	Dieser Parameter überwacht den Zustand der Eingänge des Servotreibers, der über den MECHATROLINK-II-Bus angeschlossen ist. Der Parameterwert wird bei jeder **SERVO_PERIOD** aktualisiert. Er ist ein in Bits unterteiltes Wort, das aus den in der unten stehenden Tabelle aufgelisteten Bits besteht. Die empfohlene Einstellung ist: Pn81E = 4321 & Pn511 = 654x.
Argumente	–
Beispiel	Alle Eingänge können mit diesem Wort wie folgt überwacht werden (Sigma-II-Servotreiber): **CN1-40 DRIVE_INPUTS Bit 12** **CN1-41 DRIVE_INPUTS Bit 13** **CN1-42 DRIVE_INPUTS Bit 14** **CN1-43 DRIVE_INPUTS Bit 15** **CN1-44 DRIVE_INPUTS Bit 06** **CN1-45 DRIVE_INPUTS Bit 07** **CN1-46 DRIVE_INPUTS Bit 08**
Beispiel	Alle Eingänge können mit diesem Wort wie folgt überwacht werden (Junma-Servotreiber): **CN1-1 DRIVE_INPUTS Bit 6** **CN1-2 DRIVE_INPUTS Bit 2** **CN1-3 DRIVE_INPUTS Bit 1** **CN1-4 DRIVE_INPUTS Bit 0**
Siehe auch	–

Bitnummer	Beschreibung Sigma-II	Beschreibung Junma
0	P_OT	P_OT
1	N_OT	N_OT
2	DEC Signal (wird mit Pn511.0 ausgewählt)	/DEC
3	Drehgeber Phase A	Nicht belegt
4	Drehgeber Phase B	Nicht belegt
5	Drehgeber Phase C	Nicht belegt
6	EXT1 Signal (wird mit Pn511.1 ausgewählt)	/EXT1
7	EXT2 Signal (wird mit Pn511.2 ausgewählt)	Nicht belegt
8	EXT3 Signal (wird mit Pn511.3 ausgewählt)	Nicht belegt
9	BRK-Bremsausgang	/BK
10	Reserviert	E_STP
11	Reserviert	Nicht belegt
12	IO12 (CN1-Eingangssignal in Pn81E.0 ausgewählt)	Nicht belegt
13	IO13 (CN1-Eingangssignal in Pn81E.1 ausgewählt)	Nicht belegt
14	IO14 (CN1-Eingangssignal in Pn81E.2 ausgewählt)	Nicht belegt
15	IO15 (CN1-Eingangssignal in Pn81E.3 ausgewählt)	Nicht belegt

3.2.85 DRIVE_MONITOR

Typ	Achsenparameter
Syntax	**DRIVE_MONITOR**
Beschreibung	Dieser Parameter enthält die Zustandsdaten eines Servotreibers, der über den MECHATROLINK-II-Bus an das System angeschlossen ist. Die Daten, die überwacht werden sollen, werden mit **DRIVE_CONTROL** ausgewählt und können über Trajexia Tools angezeigt oder in einem Programm verwendet werden. Die überwachten Daten werden bei jeder **SERVO_PERIOD** aktualisiert.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.86 DRIVE_READ

Typ	Achsenbefehl
Syntax	**DRIVE_READ(parameter,Groesse,VR)**
Beschreibung	Die Funktion **DRIVE_READ** liest den spezifizierten Parameter eines Servotreibers aus, der über den MECHATROLINK-II-Bus an das Trajexia-System angeschlossen ist. Nach erfolgreicher Ausführung gibt der Befehl –1 zurück und speichert den Wert an dem Speicheradresse im VR-Speicher, der durch den VR-Parameter spezifiziert ist. Wenn der Befehl nicht ausgeführt werden kann, wird der Wert 0 zurückgegeben. Der Befehl wird im Treiber für die durch **BASE** eingestellte Basisachse durchgeführt. Diese kann mit dem Modifikator **AXIS** sowie mit allen anderen Achsenbefehlen und -parametern geändert werden. Hinweis: Dieser Befehl wartet auf die Antwort von der Achse; daher kann die Ausführung des Befehls langsam sein und die Dauer variieren. Verwenden Sie diesen Befehl nicht zusammen mit anderen Befehlen, die eine schnelle Ausführung erfordern. Hinweis: Die Ausführung von **DRIVE_READ** deaktiviert zeitweilig die Anzeige des Servotreibers. Hinweis: **DRIVE_READ** gibt bei erfolgreicher Ausführung –1 zurück. Er gibt auch dann –1 zurück, wenn kein Parameter gelesen wurde, z. B. wenn die Parameternummer nicht existiert oder die falsche Größe hat.
Argumente	• **Parameter** Die Nummer des Parameters, der gelesen werden soll. Bitte beachten Sie, dass die Parameternummern hexadezimal sind. Das Format der Daten finden Sie im Servotreiber-Handbuch. • **Größe** Bei den meisten Parametern beträgt die Größe normalerweise 2 Byte. Einige spezielle Parameter können 4 Byte lang sein. Die Größen der einzelnen Parameter finden Sie im Servotreiber-Handbuch. • **VR** Die VR-Adresse, an der der gelesene Parameter bei erfolgreicher Ausführung gespeichert wird.

Beispiel	**IF DRIVE_READ($100,2,1) THEN**
	PRINT "Die Drehzahlregelkreisverstaerkung ist: ";VR(1)
	ELSE
	PRINT "Die Drehzahlregelkreisverstaerkung konnte nicht gelesen werden"
	ENDIF

Siehe auch	**DRIVE_WRITE**, **HEX**, **$ (HEXADEZIMALE EINGABE)**.

Achtung
Stellen Sie sicher, dass keine Bedienkonsole oder PC-Software an den Servotreiber angeschlossen ist, wenn dieser Befehl ausgeführt wird. Andernfalls wird die Programmausführungung unterbrochen, bis die Verbindung des anderen Gerätes mit dem Servotreiber getrennt wird.

3.2.87 DRIVE_RESET

Typ	Achsenbefehl
Syntax	**DRIVE_RESET**
Beschreibung	Der Befehl **DRIVE_RESET** setzt einen Servotreiber zurück, der über den MECHATROLINK-II-Bus angeschlossen ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

Achtung
Stellen Sie sicher, dass keine Bedienkonsole oder PC-Software an den Servotreiber angeschlossen ist, wenn dieser Befehl ausgeführt wird. Andernfalls wird die Programmausführungung unterbrochen, bis die Verbindung des anderen Gerätes mit dem Servotreiber getrennt wird.

3.2.88 DRIVE_STATUS

Typ Achsenparameter (Nur-Lesen)

Syntax **DRIVE_STATUS**

Beschreibung Bei MECHATROLINK-II-Achsen wird dieser Parameter über das STATUS-Feld im MECHATROLINK-II-Kommunikationsrahmen eingestellt und bei jedem Servozyklus aktualisiert. Diese Bits können in Trajexia Tools im Konfigurationsfenster für die intelligenten Antriebe betrachtet und in Programmen verwendet werden. Die Erläuterungen zu den einzelnen Bits finden Sie in der unten stehenden Tabelle. (Hinweis: Es sind nur die Bits aufgelistet, die für MECHATROLINK-II-Achsen relevant sind.)
Detaillierte Erklärungen zu diesen Status-Bits finden Sie im MECHATROLINK-II-Handbuch.
Bei „Flex Achsmodul"-Achsen speichert dieser Parameter den Druckmarkenstatus und Zusatzeingänge sowie die Druckmarkenauswahl. Die Erläuterungen zu den einzelnen Bits finden Sie in der zweiten Tabelle unten. (Hinweis: Es sind nur die Bits aufgelistet, die für Flex Achsmodule relevant sind.)

Argumente –

Beispiel **PRINT DRIVE_STATUS AXIS(4)**
Mit diesem Befehl wird der aktuelle Wert von **DRIVE_STATUS** für Achse (4) ausgegeben.

Beispiel
```
BASE(3)
ATYPE = 44
  IF (DRIVE_STATUS AND 32)= 32 THEN
  PRINT "REG 0 Eingang ist EIN für Achse(3)"
ENDIF
```

Siehe auch **AXIS**, **MARK**, **MARKB**, **REGIST**.

Bit	Beschreibung (MECHATROLINK-II)
0	Alarm
1	Vorsicht
2	Bereit
3	Servo ein
4	Eingeschaltet
5	Maschinensperre
6	Ausgangsposition
7	Bei Position/Drehzahl
8	Ausgabe beendet
9	Drehmomentgrenzwert
10	Erfassung abgeschlossen
11	In Bereich/Drehzahlgrenze

Bit	Beschreibung (Flex Achsmodule)
0	MARK
1	MARKB
2	REG-0 ausgewählter aktueller Wert
3	REG-1 ausgewählter aktueller Wert
4	AUX IN aktueller Wert
5	REG 0 aktueller Wert
6	REG 1 aktueller Wert

3.2.89 DRIVE_WRITE

Typ Achsenbefehl

Syntax **DRIVE_WRITE(parameter, groesse, wert [,modus])**

Beschreibung Die Funktion **DRIVE_WRITE** schreibt über den MECHATROLINK-II-Bus einen Servoparameter. Nach erfolgreicher Ausführung gibt dieser Befehl –1 zurück. Wenn der Befehl nicht ausgeführt werden kann, wird der Wert 0 zurückgegeben. Der Befehl wird im Treiber für die durch **BASE** eingestellte Basisachse durchgeführt. Diese kann mit dem Modifikator **AXIS** sowie mit allen anderen Achsenbefehlen und -parametern geändert werden. Bei einigen Parametern muss der Treiber aus- und wieder eingeschaltet werden. Der Befehl **DRIVE_RESET** kann zu diesem Zweck verwendet werden.
Hinweis: Dieser Befehl wartet auf die Antwort von der Achse; daher kann die Ausführung des Befehls langsam sein und die Dauer variieren. Verwenden Sie diesen Befehl nicht zusammen mit anderen Befehlen, die eine schnelle Ausführung erfordern.
Hinweis: Die Ausführung von **DRIVE_WRITE** deaktiviert zeitweilig die Anzeige des Servotreibers.
Hinweis: **DRIVE_WRITE** gibt bei erfolgreicher Ausführung –1 zurück. Er gibt auch dann –1 zurück, wenn kein Parameter gelesen wurde, z. B. wenn die Parameternummer nicht existiert oder die falsche Größe hat.

Argumente
- **Parameter**
  Die Nummer des Parameters, zu dem geschrieben werden soll. Bitte beachten Sie, dass die Parameternummern hexadezimal sind. Das Format der Daten finden Sie im Servotreiber-Handbuch.
- **Größe**
  Bei den meisten Parametern beträgt die Größe normalerweise 2 Byte. Einige spezielle Parameter können 4 Byte lang sein. Die Größen der einzelnen Parameter finden Sie im Servotreiber-Handbuch.
- **Wert**
  Der Wert, der in den Treiberparameter geschrieben werden soll.
- **Modus**
  Der Schreibmodus. Mögliche Werte: 0 (oder ausgelassen) – schreiben und im RAM speichern; 1 – schreiben und im EPROM speichern.

Beispiel
```
IF DRIVE_WRITE($100,2,90) THEN
  PRINT "Die neue Drehzahlregelkreisverstaerkung ist: 90"
ELSE
  PRINT "Die Drehzahlregelkreisverstaerkung konnte nicht in den RAM
  geschrieben werden"
ENDIF
```

Siehe auch • **DRIVE_READ**, **DRIVE_RESET**, **$ (HEXADEZIMALE EINGABE)**

Achtung
Stellen Sie sicher, dass keine Bedienkonsole oder PC-Software an den Servotreiber angeschlossen ist, wenn dieser Befehl ausgeführt wird. Andernfalls wird die Programmausführungung unterbrochen, bis die Verbindung des anderen Gerätes mit dem Servotreiber getrennt wird.

3.2.90 EDIT

Typ Programmbefehl

Syntax **EDIT [Zeilen_nummer]**
ED [Zeilen_nummer]

Beschreibung Der Befehl **EDIT** startet den integrierten Bildschirmeditor, mit dem ein Programm im Controller über ein VT100-Terminal modifiziert werden kann. Das aktuell ausgewählte Programm wird bearbeitet.
Die Editor-Befehle sind wie folgt:
Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann der Benutzer den Befehl aus dem Menü **Programm** auswählen.
- Editor beenden: [STRG] K und D
- Zeile löschen: [STRG] Y

Argumente
- **Zeilen_nummer**
  Die Nummer der Zeile, in der mit der Bearbeitung begonnen werden soll.

Beispiel Kein Beispiel.

Siehe auch **SELECT**.

3.2.91 ELSE

Siehe **IF..THEN..ELSE..ENDIF**.

3.2.92 ELSEIF

Siehe **IF..THEN..ELSE..ENDIF**.

3.2.93 ENCODER

Typ	Achsenparameter (Nur-Lesen)
Syntax	**ENCODER**
Beschreibung	Der Achsenparameter **ENCODER** enthält den direkten, unskalierten Wert des Drehgebers. Der Achsenparameter **MPOS** enthält die gemessene Position, die aus dem Wert von **ENCODER** automatisch berechnet wird, wobei Überläufe und Offsets berücksichtigt werden.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **MPOS**.

3.2.94 ENCODER_BITS

Typ	Achsenparameter
Syntax	**ENCODER_BITS = Wert**
Beschreibung	Dieser Achsenparameter konfiguriert die Schnittstelle für die Anzahl der Drehgeber-Bits für Flex Achsmodule SSI und EnDat Absolutwert-Drehgeberachsen. Der Parameter ist nur für Achsen mit den **ATYPE**-Werten 47 und 48 verwendbar. Bei Flex Achsmodulen mit EnDat Absolutwert-Drehgeber sind die Bits 0–7 des Parameters auf die Gesamtanzahl der Drehgeber-Bits einzustellen. Die Bits 8–14 sind auf die Anzahl der zu verwendenden Multi-Umdrehungsbits einzustellen. Bei Flex Achsmodulen mit SSI Absolutwert-Drehgeber sind die Bits 0–5 des Parameters auf die Anzahl der Drehgeber-Bits einzustellen. Bit 6 sollte 1 für binären Betrieb sein oder 0 für Gray-Code. Hinweis: Bei Verwendung eines Flex Achsmoduls als Absolutwert-Drehgeberachse ist es wichtig, diesen Parameter für die Achse zu setzen, bevor ATYPE eingestellt wird.
Argumente	–
Beispiel	**ENCODER_BITS = 25 + (256 * 12)** **ATYPE = 47** In diesem Beispiel wird ein 25-Bit EnDat-Drehgeber verwendet, der 12 Bits für Multi-Umdrehungswerte und 13 Bits pro Umdrehung hat.
Beispiel	**ENCODER_BITS = 12 + (64 * 1)** **ATYPE = 48** In diesem Beispiel wird ein 12-Bit- (4096 Positionen pro Umdrehung) SSI-Drehgeber mit binärem Ausgangstyp verwendet.
Siehe auch	**AXIS**.

3.2.95 ENCODER_CONTROL

Typ	Achsenparameter
Syntax	**ENCODER_CONTROL = Wert**
Beschreibung	Der Parameter **ENCODER_CONTROL** ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem EnDat-Absolutwert-Drehgeber mit ATYPE-Wert 47. Der Parameter steuert die Art, wie der EnDat-Drehgeber seine Position zurückmeldet. Der Drehgeber kann entweder so eingestellt werden, dass er zyklisch seine Position zurückmeldet, oder er kann auf einen Parameter-Lese-/Schreibmodus gesetzt werden. Die Standardeinstellung nach der Initialisierung ist der zyklische Rückmeldemodus. Weitere Informationen finden Sie in der Spezifikation für die EnDat Absolutwert-Drehgeberschnittstelle.
Argumente	–
Beispiel	**ENCODER_CONTROL AXIS(1) = 0** Mit diesem Befehl wird der zyklische Rückmeldemodus eingestellt.
Beispiel	**ENCODER_CONTROL AXIS(1) = 1** Mit diesen Befehl wird der Parameter-Lese-/Schreibmodus eingestellt.
Siehe auch	**AXIS**, **ENCODER**, **ENCODER_BITS**.

3.2.96 ENCODER_ID

Typ	Achsenparameter (Nur-Lesen)
Syntax	**ENCODER_ID**
Beschreibung	Dieser Parameter gibt den ID-Wert eines Absolutwert-Drehgebers für die Achse zurück. Dieser Parameter ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem Tamagawa-Absolutwert-Drehgeber mit **ATYPE**-Wert 46. Er gibt vom Drehgeber den Parameter ENID zurück, der auf 17 gesetzt ist. Weitere Informationen finden Sie in der Spezifikation für die Tamagawa-Absolutwert-Drehgeberschnittstelle. Wenn dieser Parameter auf eine Achse mit einem anderen ATYPE-Wert als 46 angewendet wird, gibt er 0 zurück.
Argumente	–
Beispiel	**>>PRINT ENCODER_ID AXIS (1)** **17.0000** Mit diesem Befehl wird der Wert für die Absolutwert-Drehgeber- ID für Achse 1 ausgegeben.
Siehe auch	**AXIS**, **ENCODER**, **ENCODER_BITS**.

3.2.97 ENCODER_RATIO

Typ	Achsenparameter
Syntax	**ENCODER_RATIO(nenner,zaehler)**
Beschreibung	Erlaubt die Skalierung der Drehgeber-Signale mit einer Nicht-Integerzahl. Dies erfolgt mit der Gleichung: **MPOS = (nenner)/(zaehler) x drehgeber_flankeneingang** Im Unterschied zu den **UNITS**-Parametern wirkt sich **ENCODER_RATIO** sowohl auf **MOVECIRC** als auch auf **CAMBOX** aus. Hinweis: Große Verhältnisse sollten vermieden werden, da sie entweder einen Verlust der Auflösung oder eine erheblich geringere Stetigkeit im Bewegungsablauf zur Folge haben. Der tatsächliche physikalische Drehgeber-Zähler ist die Grundauflösung der Achse; der Einsatz dieses Befehls kann die Möglichkeit des Motion-Controllers einschränken, alle Positionen genau zu erreichen. Hinweis: ENCODER_RATIO ersetzt nicht UNITS. Verwenden Sie **ENCODER_RATIO** nur dann, wenn es absolut notwendig ist. Verwenden Sie für die Skalierung aller anderen Achsen **UNITS**.
Argumente	• **Nenner** Eine Zahl zwischen 0 und 16777215, die dazu verwendet wird, den Nenner in der obigen Gleichung zu definieren. • **Zähler** Eine Zahl zwischen 0 und 16777215, die dazu verwendet wird, den Zähler in der obigen Gleichung zu definieren.
Beispiel	**ENCODER_RATIO(8192,7200)** **UNITS = 20** Ein Drehtisch hat einen Servomotor, der direkt mit dem Drehpunkt verbunden ist. An der Rückseite des Servomotors ist ein Drehgeber angebracht und gibt einen Wert von 8192 Zählern pro Umdrehung zurück. Die Anwendung erfordert es, dass der Tisch in Grad kalibriert ist, und zwar so, dass ein Grad durch eine Integer-Zahl von Impulsen dargestellt wird.
Siehe auch	–

3.2.98 ENCODER_READ

Typ	Achsenbefehl
Syntax	**ENCODER_READ(adresse)**
Beschreibung	Der Befehl **ENCODER_READ** ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem EnDat- Absolutwert-Drehgeber mit **ATYPE**-Wert 47. Der Parameter gibt einen 16-Bit-Drehgeberparameter zurück, der an der spezifizierten Adresse gespeichert ist. Die Bits 8–15 der Adresse sind die EnDat MRS-Feldeinstellungen; die Bits 0–7 sind der Offset innerhalb des EnDat MRS-Blocks. Wenn ein CRC-Fehler auftritt, gibt dieser Befehl –1 zurück. Weitere Informationen finden Sie in der Spezifikation für die EnDat Absolutwert-Drehgeberschnittstelle.
Argumente	• **Adresse** Spezifiziert das EnDat MRS-Feld, das gelesen werden soll.
Beispiel	**VR(100) = ENCODER_READ($A10D) AXIS(7)** Dieser Befehl liest die Anzahl der Drehgeber-Bits und speichert diesen Wert am Speicheradresse VR(100).
Siehe auch	**AXIS**, **ENCODER**, **ENCODER_BITS**.

3.2.99 ENCODER_STATUS

Typ Achsenparameter (Nur-Lesen)

Syntax **ENCODER_STATUS**

Beschreibung Dieser Parameter gibt den Zustand des Absolutwert-Drehgebers zurück. Dieser Parameter ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem Tamagawa- Absolutwert-Drehgeber mit **ATYPE**-Wert 46. Er gibt sowohl das SF-Zustandsfeld als auch das ALMC-Drehgeber-Fehlerfeld zurück. Das SF-Feld ist in den Bits 0–7, das ALMC-Feld in den Bits 8–15. Weitere Informationen finden Sie in der Spezifikation für die Tamagawa Absolutwert-Drehgeberschnittstelle.
Wenn dieser Parameter auf eine Achse mit einem anderen **ATYPE**-Wert als 46 angewendet wird, gibt er 0 zurück.

Argumente –

Beispiel **PRINT (ENCODER_STATUS AXIS (1) AND 255)**
Mit diesem Befehl wird das SF-Feld des Tamagawa-Absolutwert-Drehgebers für Achse 1 ausgegeben.

Siehe auch **AXIS**, **ENCODER**, **ENCODER_BITS**.

3.2.100 ENCODER_TURNS

Typ Achsenparameter (Nur-Lesen)

Syntax **ENCODER_TURNS**

Beschreibung Der Parameter **ENCODER_TURNS** gibt die Anzahl der Multi-Umdrehungszähler vom Drehgeber zurück.
Dieser Parameter ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem Tamagawa-Absolutwert-Geber mit **ATYPE**-Wert 46 oder einem EnDat-Absolutwert-Geber mit **ATYPE**-Wert 47.
Die Multi-Umdrehungszähler werden nach der Initialisierung nicht automatisch auf den Achsenparameter **MPOS** angewendet. Der Anwendungsprogrammierer muss sie je nach Anforderung aus dem Programm über den Befehl **OFFPOS** oder **DEFPOS** anwenden.
Wenn dieser Parameter auf eine Achse mit einem anderen **ATYPE**-Wert als 46 oder 47 angewendet wird, gibt er 0 zurück.

Argumente –

Beispiel **PRINT ENCODER_TURNS AXIS (1)**
Mit diesem Befehl wird der Multi-Umdrehungszähler des Absolutwert-Drehgebers für Achse 1 ausgegeben.

Siehe auch **AXIS**, **ENCODER**, **ENCODER_BITS**.

3.2.101 ENCODER_WRITE

Typ Achsenbefehl

Syntax **ENCODER_WRITE(adresse, wert)**

Beschreibung Der Befehl **ENCODER_WRITE** ist nur verwendbar bei Flex Achsmodulen in Verbindung mit einem EnDat-Absolutwert-Drehgeber mit **ATYPE**-Wert 47. Der Befehl schreibt einen Drehgeberparameter, der durch die Adresse spezifiziert wird. Die Bits 8–15 der Adresse sind die EnDat MRS-Feldeinstellungen; die Bits 0–7 sind der Offset innerhalb des EnDat MRS-Blocks. Wenn ein CRC-Fehler auftritt, gibt dieser Befehl 0 zurück. Beim Schreiben zu Adresse 0 wird eine Drehgeber-Rücksetzfunktion ausgeführt. Weitere Informationen finden Sie in der Spezifikation für die EnDat-Absolutwert-Drehgeberschnittstelle. Um mit diesem Befehl einen Drehgeber-Parameter erfolgreich zu schreiben, sollte der Parameter **ENCODER_WRITE** auf 1 gesetzt und der Lese-/Schreibmodus eingestellt werden.

Argumente
- **Adresse**
  Spezifiziert das EnDat MRS-Feld, zu dem geschrieben werden soll.
- **Wert**
  Ein BASIC-Ausdruck.

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **ENCODER**, **ENCODER_BITS**, **ENCODER_CONTROL**.

3.2.102 ENDIF

Siehe **IF..THEN..ELSE..ENDIF**.

3.2.103 ENDMOVE

Typ	Achsenparameter
Syntax	**ENDMOVE**
Beschreibung	Der Achsenparameter **ENDMOVE** speichert die Position am Ende der aktuellen Bewegung in Benutzereinheiten. Wenn der Achsenparameter **SERVO** aktiviert ist, kann der Parameter **ENDMOVE** geschrieben werden, um bei Achsenstillstand eine Änderung der gewünschten Position (**DPOS**) zu erreichen. Hinweis: Da die gemessene Positionen zu Anfang nicht geändert wird, sollte der Grenzwert des Schleppfehlers (**FE_LIMIT**) berücksichtigt werden. Wenn die Änderung der gewünschten Position zu groß ist, wird der Grenzwert überschritten.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **DPOS**, **FE_LIMIT**, **UNITS**.

3.2.104 EPROM

Typ	Programmbefehl
Syntax	**EPROM**
Beschreibung	Der Befehl **EPROM** speichert die im batteriegestützten RAM-Speicher des TJ1-MC__ enthaltenen BASIC-Programme im Flash-EPROM-Speicher. Ob die im Flash-EPROM-Speicher gespeicherten Programme beim Hochfahren ins RAM kopiert werden, wird durch den Systemparameter POWER_UP gesteuert. Hinweis: Trajexia Tools bietet diesen Befehl als Schaltfläche auf der Bedienkonsole an. Außerdem wird durch Pop-up-Bildschirme abgefragt, ob die Programmdaten in den Flash-Speicher geschrieben werden sollen.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**POWER_UP**, **RUNTYPE**.

3.2.105 ERROR_AXIS

Typ	Systemparameter (Nur-Lesen)
Syntax	**ERROR_AXIS**
Beschreibung	Der Achsenparameter **ERROR_AXIS** enthält die Nummer der Achse, die den Bewegungsfehler verursacht hat. Eine Bewegungsfehler tritt auf, wenn der **AXISSTATUS**-Zustand für eine der Achsen mit der Einstellung **ERRORMASK** übereinstimmt. In diesem Fall wird die Achsenfreigabe (**WDOG**) ausgeschaltet, der Parameter **MOTION_ERROR** hat den Wert 1, und der Parameter **ERROR_AXIS** enthält die Nummer der ersten Achse, bei der der Fehler auftritt.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXISSTATUS**, **ERRORMASK**, **MOTION_ERROR**, **WDOG**.

3.2.106 ERROR_LINE

Typ	Task-Parameter (Nur-Lesen)
Syntax	**ERROR_LINE**
Beschreibung	Der Parameter **ERROR_LINE** enthält die Nummer der Zeile, die den letzten BASIC-Laufzeitfehler in der Programmausführungung verursacht hat. Dieser Wert ist nur gültig, wenn der Parameter **BASICERROR TRUE** ist. Jeder Task hat seinen eigenen **ERROR_LINE**-Parameter. Verwenden Sie den Modifikator **PROC**, um auf den Parameter für einen bestimmte Task zuzugreifen. Ohne **PROC** wird der aktuelle Task ausgeführt.
Argumente	–
Beispiel	**>> PRINT ERROR_LINE PROC(4)** **23.0000**
Siehe auch	**BASICERROR**, **PROC**, **RUN_ERROR**.

3.2.107 ERRORMASK

Typ	Achsenparameter
Syntax	**ERRORMASK**
Beschreibung	Der Achsenparameter **ERRORMASK** enthält einen Maskenwert, der bei jedem Servozyklus Bit für Bit über eine **AND**-Link mit dem Achsenparameter **AXISSTATUS** verbunden ist, um zu ermitteln, ob ein Bewegungsfehler aufgetreten ist. Wenn ein Bewegungsfehler auftritt, wird die Achsenfreigabe (**WDOG**) ausgeschaltet, der Parameter **MOTION_ERROR** hat den Wert 1, und der Parameter **ERROR_AXIS** enthält die Nummer der ersten Achse, bei der der Fehler auftritt. Überprüfen Sie den Parameter **AXISVALUES** auf die Zuordnungen der Status-Bits. Die Standardeinstellung von **ERRORMASK** ist 268.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **AXISSTATUS**, **MOTION_ERROR**, **WDOG**.

Achtung
Der Benutzer muss definieren, in welchen Fällen ein Bewegungsfehler generiert wird. Für einen sicheren Betrieb wird dringend empfohlen, immer einen Bewegungsfehler zu generieren, wenn der Schleppfehler seinen Grenzwert überschritten hat.
Dies erfolgt durch Einstellung von Bit 8 von **ERRORMASK**.

3.2.108 ETHERNET

Typ	Systembefehl
Syntax	**ETHERNET(funktion, baugruppen_nummer, parameter [,werte])**
Beschreibung	Der Befehl **ETHERNET** wird dazu verwendet, bestimmte Funktionen der Ethernet-Kommunikation zu lesen und einzustellen. Der Befehl **ETHERNET** kann in der Befehlszeile über die serielle Schnittstelle 0 eingegeben werden, wenn Trajexia Tools vom System getrennt ist. Hinweis: Um die neuen Parameter zu aktivieren, müssen Sie die Versorgungsspannung für Trajexia aus- und wieder einschalten.
Argumente	• **Funktion** 0 = Lesen, 1 = Schreiben • **Baugruppen_nummer** –1 • **Parameter** 0 = IP-Adresse; 2 = Subnetzmaske; 3 = MAC-Adresse; 8 = Gateway; 11 = ARP-Cache (Nur-Lesen) • **Werte** Der für einen Schreibvorgang erforderliche Parameter
Beispiel	**ETHERNET(1,–1,0,192,200,185,2)** Stellen Sie die Trajexia-IP-Adresse auf 192.200.185.002 ein.
Siehe auch	–

3.2.109 EX

Typ	Systembefehl
Syntax	**EX[(option)]**
Beschreibung	Setzt den Controller in den Zustand wie beim Hochfahren zurück. Mit dem Befehl **EX** kann die Rücksetzung und auf zwei verschiedene Arten erfolgen. **EX** ohne das Argument oder **EX(0)**, um die Software des Controllers zurückzusetzen. **EX(1)** führt eine Hardware-Rücksetzung des Controllers aus.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.110 EXP

Typ	Mathematische Funktion
Syntax	**EXP(ausdruck)**
Beschreibung	Die Funktion **EXP** gibt den Exponentialwert eines Ausdrucks zurück.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> print exp(1.0)** **2.7183**
Siehe auch	–

3.2.111 FALSE

Typ	Konstante (Nur-Lesen)
Syntax	**FALSE**
Beschreibung	Die Konstante **FALSE** gibt den numerischen Wert 0 zurück.
Argumente	–

Beispiel	**Test:**
	res = IN(0) OR IN(2)
	IF res = FALSE THEN
	PRINT "Eingaenge sind aus"
	ENDIF

Siehe auch	–

3.2.112 FAST_JOG

Typ	Achsenparameter
Syntax	**FAST_JOG**
Beschreibung	Der Achsenparameter **FAST_JOG** enthält die Nummer des Eingangs, der als Schnelltippeingang verwendet werden soll. Die Nummer kann eine Zahl zwischen 0 und 7 sein. Standardmäßig ist der Parameter auf –1 eingestellt, und es ist kein Eingang ausgewählt. Der Schnelltippeingang steuert die Tippgeschwindigkeit zwischen zwei Drehzahlen. Wenn der Schnelltippeingang eingestellt ist, wird die durch den Achsenparameter **SPEED** vorgegebene Drehzahl als Schrittgeschwindigkeit verwendet. Wenn der Schnelltippeingang nicht eingestellt ist, wird die durch den Achsenparameter **JOGSPEED** vorgegebene Drehzahl verwendet. Hinweis: Dieser Eingang ist in aktiviertem Zustand null.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **FWD_JOG**, **JOGSPEED**, **REV_JOG**, **SPEED**.

3.2.113 FASTDEC

Typ	Achsenparameter
Syntax	**FASTDEC**
Beschreibung	Die Standardeinstellung ist null. Wenn für **FASTDEC** ein anderer Wert als Null spezifiziert ist, verzögert die Achse bei dieser Verzögerungsrate vom „aktuellen Wert bis Null“, wenn ein Achsen-Begrenzungsschalter oder eine Grenzposition erreicht ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.114 FE

Typ	Achsenparameter (Nur-Lesen)
Syntax	**FE**
Beschreibung	Der Achsenparameter **FE** enthält den Positionsfehler in Benutzereinheiten. Dieser wird berechnet aus der gewünschten Position (achsenparameter **DPOS**) minus die gemessene Position (achsenparameter **MPOS**). Der Wert des Schleppfehlers kann mit den Achsenparametern **FE_LIMIT** und **FE_RANGE** überprüft werden.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **DPOS**, **FE_LIMIT**, **FE_RANGE**, **MPOS**, **UNITS**.

3.2.115 FE_LATCH

Typ	Achsenparameter (Nur-Lesen)
Syntax	**FE_LATCH**
Beschreibung	Enthält den ersten **FE**-Wert, der bewirkt hat, dass die Achse den Controller in den Zustand MOTION_ERROR geschaltet hat. Dieser Wert wird nur gesetzt, wenn der **FE** den Grenzwert **FE_LIMIT** überschreitet und der Parameter **SERVO** auf 0 gesetzt wurde. **FE_LATCH** wird auf 0 zurückgesetzt, wenn der **SERVO**-Parameter der Achse wieder auf 1 gesetzt ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.116 FE_LIMIT

Typ	Achsenparameter
Syntax	**FE_LIMIT**
Beschreibung	Der Achsenparameter **FE_LIMIT** enthält den Grenzwert für den maximal zulässigen Schleppfehler in Benutzereinheiten. Bei einer Überschreitung wird Bit 8 des Parameters **AXISSTATUS** der Achse gesetzt. Wenn der Parameter **ERRORMASK** richtig gesetzt ist, wird ein Bewegungsfehler generiert. Dieser Grenzwert dient zum Schutz gegen Fehlerzustände, z. B. mechanische Sperrung, Verlust der Rückmeldung vom Drehgeber usw.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **AXISSTATUS**, **ERRORMASK**, **FE**, **FE_RANGE**, **UNITS**.

3.2.117 FE_LIMIT_MODE

Typ	Achsenparameter
Syntax	**FE_LIMIT_MODE = Wert**
Beschreibung	Wenn dieser Parameter auf null gesetzt ist, löst die Achse sofort einen **MOTION_ERROR** aus, wenn der **FE** den Wert für **FE_LIMIT** überschreitet. Wenn **FE_LIMIT_MODE** auf 1 gesetzt ist, generiert die Achse nur dann einen **MOTION_ERROR**, wenn der **FE** den Wert für **FE_LIMIT** während zwei aufeinander folgender Servozyklen überschreitet. Das heißt: Wird **FE_LIMIT** nur bei einem Servozyklus überschritten, wird dies ignoriert. Der Standardwert für **FE_LIMIT_MODE** ist 0.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.118 FE_RANGE

Typ	Achsenparameter
Syntax	**FE_RANGE**
Beschreibung	Der Achsenparameter **FE_RANGE** enthält den Grenzwert für den Schleppfehler-Warnbereich in Benutzereinheiten. Wenn der Schleppfehler diesen Wert an einer Servoachse übersteigt, wird Bit 1 in dem Achsenparameter **AXISSTATUS** eingeschaltet. Dieser Bereich wird als erstes Zeichen für Fehlerzustände in der Anwendung benutzt (vergleiche **FE_LIMIT**).
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **AXISSTATUS**, **ERRORMASK**, **FE**, **UNITS**.

3.2.119 FHOLD_IN

Typ	Achsenparameter
Syntax	**FHOLD_IN** **FH_IN**
Beschreibung	Der Achsenparameter **FHOLD_IN** enthält die Nummer des Eingangs, der als Feedhold-Eingang verwendet werden soll. Der gültige Eingangsbereich ist 0 bis 31. Die Werte 0 bis 15 stellen physikalisch die Eingänge des TJ1-MC__ E/A-Steckverbinders dar und werden für alle Achsen gemeinsam benutzt. Die Werte 16 bis 27 stellen Software-Eingänge dar, die beliebig in Programmen und Befehlen verwendet werden können, z. B. IN und OP. Sie gelten ebenfalls für alle Achsen. Die Werte 28 bis 31 werden direkt den am CN1-Steckverbinder vorhandenen Treiber-Eingängen zugewiesen und sind für jede Achse eindeutig. Welche Treiber-Eingänge den Eingängen 28 bis 31 zugewiesen werden, hängt von der Servotreiber-Einstellung in Parameter Pn81E ab. Die empfohlene Einstellung ist Pn81E = 0 x 4321, mit folgender Zuordnung: Standardmäßig ist der Parameter auf –1 eingestellt, und es ist kein Eingang ausgewählt. Hinweis: Dieser Eingang ist in aktiviertem Zustand null.
Sigma II	• Eingang 28: CN1-40 • Eingang 29: CN1-41 • Eingang 30: CN1-42 • Eingang 31: CN1-43
Sigma III	• Eingang 28: CN1-13 • Eingang 29: CN1-7 • Eingang 30: CN1-8 • Eingang 31: CN1-9
Junma	• Eingang 26: CN1-2 • Eingang 27: CN1-1
	Weitere Informationen zum Einstellen des Parameters Pn81E finden Sie im Servotreiber-Handbuch. Standardmäßig ist der Parameter auf –1 eingestellt, und es sind keine Eingänge ausgewählt.

Wenn eine Eingangsnummer gesetzt ist und der Feedhold-Eingang aktiviert wird, ändert sich die Drehzahl der Achse zu dem Wert, der im Achsenparameter **FHSPEED** eingestellt ist. Die aktuelle Bewegung wird nicht abgebrochen. Außerdem wird Bit 7 des Parameters **AXISSTATUS** gesetzt. Wenn der Eingang wieder zurückgesetzt wird, wird jede Bewegung, die beim Setzen des Eingangs ausgeführt wurde, mit der programmierten Drehzahl fortgesetzt. Hinweis: Diese Funktion ist nur bei Bewegungen im Drehzahlbetrieb möglich. Auf Bewegungen, die nicht im Drehzahlbetrieb ausgeführt werden (**CAMBOX**, **CONNECT** und **MOVELINK**), wirkt sich die Funktion nicht aus.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **AXISSTATUS**, **UNITS**.

3.2.120 FHSPEED

Typ Achsenparameter

Syntax **FHSPEED**

Beschreibung Der Achsenparameter **FHSPEED** enthält die Feedhold-Geschwindigkeit. Dieser Parameter kann auf den Wert in Benutzereinheiten/s gesetzt werden, mit dem sich die Achse bewegt, wenn der Feedhold-Eingang aktiviert wird. Die aktuelle Bewegung wird nicht abgebrochen. **FHSPEED** kann jeden positiven Wert einschließlich 0 annehmen. Der Standardwert ist 0. Hinweis: Diese Funktion ist nur bei Bewegungen im Drehzahlbetrieb möglich. Auf Bewegungen, die nicht im Drehzahlbetrieb ausgeführt werden (**CAMBOX**, **CONNECT** und **MOVELINK**), wirkt sich die Funktion nicht aus.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **AXISSTATUS**, **FHOLD_IN**, **UNITS**.

3.2.121 FINS_COMMS

Typ Kommunikationsbefehl

Syntax **FINS_COMMS(typ, netzwerk, teilnehmer, baugruppe, ziel_bereich, ziel_offset, laenge, lokaler_bereich, lokaler_offset, zeitueberschreitung[, ip1, ip2, ip3, ip4])**

Beschreibung FINS (Factory Interface Network Service) ist ein proprietäres OMRON-Kommunikationsprotokoll. Ein Teilbereich dieses Protokolls ist in Trajexia implementiert. Das FINS-Protokoll wurde in der Absicht implementiert, eine nahtlose Kommunikation mit anderen OMRON-Geräten (SPS, MMS, usw.) und Software (CX-Drive, CX-Server, usw.) zu ermöglichen. Weitere Informationen über das FINS-Kommunikationsprotokoll finden Sie in Abschnitt 4.2.4 und im Referenzhandbuch für Kommunikationsbefehle, Kat.-Nr. W342-E1, Abschnitte 3 und 5.
Trajexia verfügt über integrierte FINS-Client-Funktionen, damit es die FINS-Kommunikation mit FINS-Slave-Geräten über **FINS_COMMS** initiieren kann. Sowohl die FINS 0101- (Speicher lesen) als auch die FINS 0102- (Speicher schreiben) Befehle sind implementiert. Mit FINS 0101 kann der Speicher aus anderen Geräten mit FINS-Serverfähigkeit gelesen werden. FINS 0102 kann zum Schreiben von Daten zu Geräten mit FINS-Serverfähigkeit verwendet werden. Dieser Befehl gibt einen der folgenden Werte zurück, abhängig vom Ergebnis der Ausführung:
–1: Der Befehl wurde erfolgreich ausgeführt.
0: Der Befehl konnte nicht ausgeführt werden.
1: Die Abfrage wurde nicht gesendet, weil der Client oder das FINS-Protokoll belegt war.
2: Einer oder mehrere der Abfrageparameter ist/sind ungültig.
3: Ungültiger Quell-Speicherbereich.
4: Abfrage wurde gesendet, aber keine Antwort vom dezentralen Server innerhalb des Zeitüberschreitungs-Zeitraums empfangen.
5: Fehlerantwort-Code vom dezentralen Server empfangen.

Argumente

- **Typ**
  Der Typ des FINS-Befehls. 0 bedeutet FINS 0101, Speicher aus Ziel-FINS-Server lesen. 1 bedeutet FINS 0102, Speicher zu Ziel-FINS-Server schreiben.
- **Netzwerk**
  Das Zielnetzwerk. Weitere Angaben finden Sie im Referenzhandbuch für Kommunikationsbefehle, Kat.-Nr. W342-E1, Abschnitt 3.
- **Teilnehmer**
  Der Teilnehmer des Ziel-FINS-Servers. Weitere Angaben finden Sie im Referenzhandbuch für Kommunikationsbefehle, Kat.-Nr. W342-E1, Abschnitt 3.
- **Baugruppe**
  Die Baugruppennummer des Ziel-FINS-Servers. Weitere Angaben finden Sie im Referenzhandbuch für Kommunikationsbefehle, Kat.-Nr. W342-E1, Abschnitt 3.
- **Ziel_Bereich**
  Der Speicherbereich, auf den auf dem-FINS-Zielserver zugegriffen wird. Bereich: 128..255. Bitte beachten Sie, dass dieser Bereich einen der folgenden Werte haben muss, wenn das Ziel ein anderes Trajexia-System ist: 0xB0: Integer-VR-Wert; 0x82: Integer-TABLE-Wert; 0xC2: Gleitkomma-TABLE-Wert.
- **Ziel_Offset**
  Der Speicher-Offset auf dem FINS-Zielserver. Bereich: 0..65535. Bitte beachten Sie, dass die Grenzwerte für die TABLE- oder VR-Adressen für diesen Bereich eingeschränkter sind, wenn das Ziel ein anderes Trajexia-System ist.
- **Länge**
  Die Anzahl der zu übertragenden Elemente. Die Anzahl richtet sich nach der FINS-Rahmenlänge und den Fähigkeiten des Clients und des Ziel-Servers. Bei einem Trajexia-System sind 1 ... 700 Integer-Werte oder 1 … 350 Gleitkommawerte möglich.
- **lokaler_Bereich**
  Der lokale (Quell-) Speicherbereich. Bitte beachten Sie, dass dieser Bereich einen der folgenden Werte haben muss, wenn das Ziel ein anderes Trajexia-System ist: 0x00: Integer-VR-Wert; 0x01: Integer-TABLE-Wert; 0x02: Gleitkomma-TABLE-Wert.
- **lokaler_Offset**
  Der Offset des ersten Wertes im lokalen (Quell-) Speicherbereich. Der Bereich richtet sich nach der Größe des VR- oder TABLE-Arrays und dem Wert für das Längenargument.
- **Zeitüberschreitung**
  Die Anzahl der Millisekunden, die auf eine Antwort vom FINS-Zielserver gewartet werden muss, bevor die Zeit überschritten ist.
- **IP1, IP2, IP3, IP4**
  Optionale Parameter, die die (Ziel-) Server-IP-Adresse definieren. Diese Argumente müssen benutzt werden, wenn sowohl das Trajexia-System als auch der FINS-Zielserver nicht zu demselben Netzwerk gehören.

Beispiel

Ein Trajexia-System und eine OMRON CJ1-SPS mit Ethernet-Baugruppensystem CJ1W-ETN11 sind mit demselben Netzwerk verbunden. Die IP-Adresse des Trajexia-Systems ist 192.168.0.5. Die IP-Adresse der SPS-Ethernet-Baugruppe ist 192.168.0.12.
Wenn der Befehl **FINS_COMMS(0,0,12,0,$82, 1000,20,0,500,5000,192,168,0,12)** ausgeführt wird, werden 20 Worte (**Länge = 20**) des DM-SPS-Speicherbereichs (**Ziel_Bereich = $82**) ab DM1000 ausgelesen (**Ziel_Offset = 1000**) und im Integer-Format in den Trajexia-VR-Speicher geschrieben (**lokaler_Bereich = 0**), und zwar ab VR(500) (**lokaler_Offset = 500**). Daher werden die Werte DM1000 bis DM1019 im SPS-Speicherbereich im Trajexia-Speicher von VR(500) bis VR(519) abgelegt. Die Zeitüberschreitung ist auf 5 Sekunden eingestellt.
Wenn der Befehl **FINS_COMMS(1,0,12,0,$80, 50,10,0,300,3000,192,168,0,12)** ausgeführt wird, werden 10 Worte (**Länge = 10**) des Trajexia-VR-Speichers als Integer (**lokaler_Bereich = 0**) ab VR(300) (**lokaler_Offset = 300**) in den CIO-Bereich der SPS geschrieben (**Ziel_Bereich = $80**), und zwar ab CIO50 (**Ziel_Offset = 50**). Daher werden die Werte VR(300) bis VR(309) im Trajexia -Speicherbereich im Speicher der SPS von CIO50 bis CIO59 abgelegt.
Die Zeitüberschreitung ist auf 3 Sekunden eingestellt.

Siehe auch

–

3.2.122 FLAG

Typ Systembefehl

Syntax **FLAG(merker_nummer [,wert])**

Beschreibung Der Befehl **FLAG** wird dazu verwendet, eine Bank von 32 Merker-Bits einzustellen und zu lesen. Der Befehl **FLAG** kann zusammen mit einem oder zwei Parametern verwendet werden. Mit einem spezifizierten Parameter wird der Zustand des gegebenen Merker-Bits zurückgegeben. Mit zwei spezifizierten Parametern wird das gegebene Merker-Bit auf den Wert des zweiten Parameters gesetzt. Der Befehl **FLAG** dient dazu, die Kompatibilität mit älteren Controllern zu unterstützen und ist für neue Programme nicht empfehlenswert.

Argumente
- **Merker_nummer**
  Die Merkernummer ist ein Wert von 0..31.
- **Wert**
  Wird dieser Parameter spezifiziert, gibt er den Zustand des gegebenen Merkers an, d. h. Ein oder Aus. Die Schreibweise kann auch 1 oder 0 sein.

Beispiel **FLAG(27,ON)**
Setzt Merker-Bit 27 auf Ein.

Siehe auch –

3.2.123 FLAGS

Typ Systembefehl

Syntax **FLAGS([wert])**

Beschreibung Liest und setzt **FLAGS** als Block. Der Befehl **FLAGS** dient dazu, die Kompatibilität mit älteren Controllern zu unterstützen und ist für neue Programme nicht empfehlenswert. Die 32 Merker-Bits können mit **FLAGS** ausgelesen und mit **FLAGS(wert)** eingestellt werden.

Argumente
- **Wert**
  Die dezimale Entsprechung des Bit-Musters, auf das die Merker eingestellt werden müssen. Siehe Tabelle unten.

Beispiel **FLAGS(146) ' 2 + 16 + 128**
Setzt Merker 1,4 und 7 auf EIN, alle anderen auf AUS.

Beispiel **IF (FLAGS und 8) <>0 dann GOSUB irgendwo**
Test, ob Merker 3 gesetzt ist.

Siehe auch –

Bitnummer	Dezimalwert
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

3.2.124 FOR..TO..STEP..NEXT

Typ Programmsteuerungsbefehl

Syntax
```
FOR Variable = Start TO Ende [STEP Inkrement]
  Befehle
NEXT Variable
```

Beschreibung Die **FOR ... NEXT** Schleife erlaubt es, das Programmsegment zwischen **FOR** und der **NEXT**-Anweisung mehrmals zu wiederholen.
Am Eingang der Schleife wird die Variable auf den Startwert initialisiert, und der Befehlsblock wird anschließend ausgeführt. Bei Erreichen des Befehls **NEXT** wird die Variable um das Inkrement erhöht, das nach **STEP** spezifiziert ist. Der **STEP**-Wert kann positiv oder negativ sein; bei Auslassung wird der Wert 1 angenommen.
Solange die Variable kleiner oder gleich dem Endwert ist, wird der Befehlsblock wiederholt ausgeführt, bis die Variable größer als der Endwert ist. In diesem Fall fährt die Programmausführung nach **NEXT** fort.
Hinweis: **FOR ... NEXT** -Anweisungen können in einem BASIC-Programm bis zu acht Ebenen tief verschachtelt werden.

Argumente
- **Variable**
  Ein BASIC-Ausdruck.
- **Start**
  Ein BASIC-Ausdruck.
- **Ende**
  Ein BASIC-Ausdruck.
- **Inkrement**
  Ein BASIC-Ausdruck.
- **Befehle**
  Einer oder mehrere BASIC-Befehle.

Beispiel
```
FOR opnum = 8 TO 13
  OP(opnum,ON)
NEXT opnum
```
Diese Schleife schaltet die Ausgänge 8 bis 13 ein.

Beispiel
```
schleife:
  FOR Dist = 5 TO –5 STEP –0,25
    MOVEABS(dist)
    GOSUB auf_nehmen
  NEXT Dist
```
Das Inkrement **STEP** kann positiv oder negativ sein.

Beispiel
```
Schleife1:
  FOR I1 = 1 TO 8
  Schleife2:
    FOR I2 = 1 TO 6
      MOVEABS(I1 * 100,I2 * 100)
      GOSUB 1000
    NEXT I2
  NEXT I1
```
FOR..TO..STEP..NEXT können verschachtelt werden (bis zu acht Ebenen tief), wenn die inneren **FOR** und **NEXT** -Befehle beide innerhalb der äußeren **FOR..TO..STEP..NEXT** -Schleife sind.

Siehe auch **REPEAT..UNTIL**, **WHILE..WEND**.

3.2.125 FORWARD

Typ	Achsenbefehl
Syntax	**FORWARD** **FO**
Beschreibung	Der Befehl **FORWARD** bewegt eine Achse kontinuierlich mit der Geschwindigkeit vorwärts, die im Achsenparameter **SPEED** eingestellt ist. Die Beschleunigungsrate ist im Achsenparameter **ACCEL** definiert. **FORWARD** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Hinweis: Die Vorwärtsbewegung kann mit dem Befehl **CANCEL** oder **RAPIDSTOP** oder durch Erreichen des Vorwärts-Grenzwertes gestoppt werden.
Argumente	–
Beispiel	**Start:** **FORWARD** **WAIT UNTIL IN(0) = ON ' Warte auf Stoppsignal** **CANCEL**
Siehe auch	**AXIS**, **CANCEL**, **RAPIDSTOP**, **REVERSE**, **UNITS**.

3.2.126 FPGA_VERSION

Typ	Slot-Parameter
Syntax	**FPGA_VERSION SLOT(baugruppen_nummer)**
Beschreibung	Dieser Parameter gibt mit Baugruppen_nummer die FPGA-Version der Baugruppe in einem Controller-System zurück.
Argumente	• **Baugruppen_nummer** Die Baugruppennummern sind –1 bis 6 einschließlich 0, wobei –1 der TJ1-MC__ und 0 die Einheit unmittelbar rechts vom TJ1-MC__ ist.
Beispiel	–
Siehe auch	–

3.2.127 FRAC

Typ	Mathematische Funktion
Syntax	**FRAC(ausdruck)**
Beschreibung	Die Funktion **FRAC** gibt den Dezimalbruchteil des Ausdrucks zurück.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT FRAC(1.234)** **0.2340**
Siehe auch	–

3.2.128 FRAME

Typ	Systemparameter
Syntax	**FRAME = Wert**
Beschreibung	Wird verwendet, um den Rahmen zu spezifizieren, innerhalb dessen bei Rahmentransformationen operiert werden soll. Rahmentransformationen dienen dazu, Bewegungen in einem Koordinaten-Bezugsrahmen mit mehreren Achsen zu ermöglichen, wobei die Bewegungen nicht eins-zu-eins mit den Achsen übereinstimmen. Ein Beispiel ist ein SCARA-Robotarm mit Gelenkachsen. Damit die Spitze des Robotarms geradlinige Bewegungen in X-Y-Richtung ausführen kann, müssen die Motoren die Bewegungen in einem Muster ausführen, das durch die Geometrie des Roboters bestimmt wird. Rahmentransformationen zur Ausführung von Funktionen wie dieser müssen aus der Quellsprache C kompiliert und in die Controller-Systemsoftware geladen werden. Wenden Sie sich an OMRON, wenn Sie solche Funktionen benötigen. Ein Maschinensystem kann mit mehreren verschiedenen Rahmen spezifiziert werden. Der aktuell aktive **FRAME** wird mit dem Systemparameter **FRAME** spezifiziert. Standardmäßig ist **FRAME** 0, was einer Eins-zu-eins-Transformation entspricht.
Argumente	–
Beispiel	**FRAME = 1**
Siehe auch	–

3.2.129 FREE

Typ Systemfunktion

Syntax **FREE**

Beschreibung Die Funktion **FREE** gibt den freien Speicherplatz zurück, der für Anwenderprogramme und TABLE-Array-Elemente verfügbar ist.
Hinweis: Jede Zeile beansprucht mindestens 4 Zeichen (Byte) im Speicher. Diese sind für die Länge dieser Zeile, die Länge der vorherigen Zeile, die Anzahl der Leerzeichen am Anfang der Zeile und ein einzelnes Befehls-Token. Weitere Befehle benötigen ein Byte pro Token; die meisten Daten sind im ASCII-Format gespeichert.
Der TJ1-MC__ kompiliert Programme, bevor sie ausgeführt werden, d. h., es wird der doppelte Speicherplatz benötigt, um ein Programm ausführen zu können.

Argumente –

Beispiel
```
>> PRINT FREE
47104.0000
```

Siehe auch –

3.2.130 FS_LIMIT

Typ Achsenparameter

Syntax **FS_LIMIT**
FSLIMIT

Beschreibung Der Achsenparameter **FS_LIMIT** enthält die Absolutposition des Vorwärts-Softwaregrenzwertes in Benutzereinheiten.
Ein Softwaregrenzwert für die Vorwärtsbewegung kann aus dem Programm heraus gesetzt werden, um den Arbeitsbereich der Maschine zu steuern. Wenn der Grenzwert erreicht ist, verzögert der TJ1-MC__ auf 0, dann wird die Bewegung abgebrochen. Bit 9 im Achsenparameter **AXISSTATUS** wird eingeschaltet, wenn die Achsenposition größer ist als **FS_LIMIT**.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **AXISSTATUS**, **UNITS**.

3.2.131 FWD_IN

Typ Achsenparameter

Syntax **FWD_IN**

Beschreibung Der Achsenparameter **FWD_IN** enthält die Nummer des Eingangs, der als Vorwärts-Endlage verwendet werden soll. Der gültige Eingangsbereich ist 0 bis 31. Die Werte 0 bis 15 stellen physikalisch die Eingänge des TJ1-MC__ E/A-Steckverbinders dar und werden für alle Achsen gemeinsam benutzt. Die Werte 16 bis 27 stellen Software-Eingänge dar, die beliebig in Programmen und Befehlen verwendet werden können, z. B. IN und OP. Sie gelten ebenfalls für alle Achsen. Die Werte 28 bis 31 werden direkt den am CN1-Steckverbinder vorhandenen Treiber-Eingängen zugewiesen und sind für jede Achse eindeutig. Welche Treiber-Eingänge den Eingängen 28 bis 31 zugewiesen werden, hängt von der Servotreiber-Einstellung in Parameter Pn81E ab. Die empfohlene Einstellung ist Pn81E = 0 x 4321, mit folgender Zuordnung:
Wenn eine Eingangsnummer gesetzt und der Grenzwert erreicht ist, wird jede Vorwärtsbewegung dieser Achse gestoppt. Bit 4 von **AXISSTATUS** wird ebenfalls gesetzt.
Hinweis: Dieser Eingang ist in aktiviertem Zustand null.

Sigma II
- Eingang 28: CN1-40
- Eingang 29: CN1-41
- Eingang 30: CN1-42
- Eingang 31: CN1-43

Sigma III
- Eingang 28: CN1-13
- Eingang 29: CN1-7
- Eingang 30: CN1-8
- Eingang 31: CN1-9

Junma
- Eingang 26: CN1-2
- Eingang 27: CN1-1

Weitere Informationen zum Einstellen des Parameters Pn81E finden Sie im Servotreiber-Handbuch. Standardmäßig ist der Parameter auf –1 eingestellt, und es sind keine Eingänge ausgewählt.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **AXISSTATUS REV_IN**.

3.2.132 FWD_JOG

Typ Achsenparameter

Syntax **FWD_JOG**

Beschreibung Der Achsenparameter **FWD_JOG** enthält die Nummer des Eingangs, der als Jog-Vorwärtseingang verwendet werden soll. Die Nummer kann eine Zahl zwischen 0 und 7 sein. Standardmäßig ist der Parameter auf –1 eingestellt, und es ist kein Eingang ausgewählt. Hinweis: Dieser Eingang ist in aktiviertem Zustand null.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **FAST_JOG**, **JOGSPEED**, **REV_JOG**.

3.2.133 GET

Typ E/A-Befehl

Syntax **GET [#n,] Variable**

Beschreibung Der Befehl **GET** ordnet den ASCII-Code eines empfangenen Zeichens einer Variablen zu. Wenn der Eingangsspeicher der seriellen Schnittstelle leer ist, wird die Programmausführung angehalten, bis ein Zeichen empfangen worden ist. Die Kanäle 5 bis 7 sind logische Kanäle, die bei der Verwendung von Trajexia Tools der Programmierschnittstelle 0 übergeordnet sind. Hinweis: Kanal 0 ist für die Verbindung zu Trajexia Tools und/oder der Befehlszeilenschnittstelle reserviert. Bitte achten Sie darauf, dass es bei diesem Kanal für diese Funktion Probleme geben kann.

Argumente
- **n**
  Das spezifizierte Eingabegerät. Wird dieses Argument ausgelassen, wird die Schnittstelle verwendet, die durch **INDEVICE** spezifiziert ist. Siehe Tabelle unten.
- **Variable**
  Der Name der Variablen, die den ASCII-Code empfangen soll.

Beispiel **GET#5, k**
Diese Zeile speichert das über Kanal 5 der Trajexia Tools-Schnittstelle empfangene ASCII-Zeichen in **k**.

Siehe auch **INDEVICE**, **INPUT**, **KEY**, **LINPUT**

Nummer des Eingabegerätes	Beschreibung
0	Programmierschnittstelle 0
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.134 GLOBAL

Typ Systembefehl

Syntax **GLOBAL "Name", Vr_nummer**

Beschreibung Legt den Namen als Referenz für eine der globalen VR-Variablen fest. Der Name kann dann sowohl innerhalb des Programms, das die Definition für **GLOBAL** enthält als auch von allen anderen Programmen im Trajexia Tools 2-Projekt verwendet werden.
Hinweis: Das Programm mit der Definition für **GLOBAL** muss zuerst laufen, bevor die Bezeichnung in anderen Programmen verwendet wird. Außerdem sollte nur dieses Programm zu dem Zeitpunkt laufen, zu dem GLOBAL ausgeführt wird; andernfalls wird eine Fehlermeldung angezeigt, und das Programm stoppt, wenn versucht wird, diesen Befehl auszuführen. Für einen schnellen Start sollte das Programm außerdem der einzige Prozess sein, der beim Hochfahren läuft.
Wenn GLOBAL definiert wird, bleibt die Definition bis zum nächsten Zurücksetzen des TJ1-MC__ aktiv; dies kann erfolgen, indem die Versorgungsspannung aus und wieder eingeschaltet wird oder indem der EX-Befehl ausgeführt wird.
In Programmen, die den definierten Befehl **GLOBAL** benutzen, hat **Name** die gleiche Bedeutung wie **VR(vr_number)**.
Verwenden Sie nicht die Syntax: **VR(name)**.
Es können maximal 128 **GLOBAL**-Befehle definiert werden.

Argumente
- **Name**
  Jeder benutzerdefinierte Name, der alphanumerische oder Unterstrich-Zeichen enthält.
- **VR_nummer**
  Die Nummer der VR, die mit **Name** assoziiert ist.

Beispiel
```
GLOBAL "Gewinde_steigung",12
GLOBAL "verhaeltnis1",534
verhaeltnis1 = 3,56
Gewinde_steigung = 23,0
PRINT Gewinde_steigung, verhaeltnis1
```

Siehe auch –

3.2.135 GOSUB..RETURN

Typ Programmsteuerungsbefehl

Syntax
```
GOSUB Kennzeichnung
...
RETURN
```

Beschreibung Die Struktur **GOSUB** ermöglicht den Sprung in ein Unterprogramm-. **GOSUB** speichert die Position der Zeile nach dem **GOSUB**-Befehl und springt dann zu der spezifizierten Bezeichnung. Beim Erreichen der Anweisung **RETURN** kehrt die Programmausführung zu der gespeicherten Position zurück.
Hinweis: Unterprogramme können in jedem Task in bis zu 8 Ebenen verschachtelt werden.

Argumente
- **label**
  Eine gültiger Label, der im Programm vorkommt. Bei einem ungültigen Label wird vor der Ausführung ein Kompilierungsfehler ausgegeben. Labels können Zeichenketten beliebiger Länge sein, aber nur die ersten 15 Zeichen werden ausgewertet.

Beispiel
```
haupt:
  GOSUB unterprogramm
  GOTO haupt
unterprogramm:
  PRINT "Gemessene Position = ";MPOS;CHR(13);
  RETURN
```

Siehe auch **GOTO**

3.2.136 GOTO

Typ Programmsteuerungsbefehl

Syntax **GOTO label**

Beschreibung Die Struktur **GOTO** ermöglicht einen Sprung in der Programmausführung. **GOTO** bewirkt, dass die Programmausführung zu der Programmzeile springt, die den Label enthält.

Argumente
- **label**
  Eine gültiger Label, der im Programm vorkommt. Bei einem ungültigen Label wird vor der Ausführung ein Kompilierungsfehler ausgegeben. Labels können Zeichenketten beliebiger Länge sein, aber nur die ersten 15 Zeichen werden ausgewertet

Beispiel
```
schleife:
  PRINT "Gemessene Position = ";MPOS;CHR(13);
  GOTO schleife
```

Siehe auch **GOSUB..RETURN**

3.2.137 HALT

Typ Systembefehl

Syntax **HALT**

Beschreibung Der Befehl **HALT** stoppt die Ausführung aller aktuell laufenden Programme. Der Befehl kann in der Befehlszeile und in Programmen verwendet werden. Der Befehl **STOP** kann zum Stoppen eines einzelnen Programms verwendet werden.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **PROCESS**, **STOP**.

3.2.138 HEX

Typ E/A-Befehl

Syntax **HEX**

Beschreibung Dieser Befehl wird in einer Druckanweisung verwendet, um eine Zahl im Hexadezimalformat auszugeben.

Argumente –

Beispiel
```
PRINT#5,HEX(IN(8,16))
```

Siehe auch –

3.2.139 HLM_COMMAND

Typ Kommunikationsbefehl

Syntax **HLM_COMMAND(befehl, schnittstelle [, teilnehmer [, mc_bereich/betriebsart [, mc_offset]]])**

Beschreibung Der Befehl **HLM_COMMAND** führt einen Host-Link-Befehl bei einem oder allen Host-Link-Slaves an der ausgewählten Schnittstelle durch. Die Programmausführung wird angehalten, bis die Antwort-Zeichenkette empfangen wurde oder eine Zeitüberschreitung eingetreten ist. Die Zeitüberschreitung wird durch den Parameter **HLM_TIMEOUT** spezifiziert. Der Status der Übertragung kann mit den Parameter **HLM_STATUS** überwacht werden.
Hinweise:
- Wenn Sie den Befehl **HLM_READ** benutzen, stellen Sie sicher, dass das Host-Link-Masterprotokoll eingerichtet ist, indem Sie den Befehl **SETCOM** verwenden.
- Die Host-Link-Masterbefehle dürfen nur von einem Programm ausgeführt werden, um Timing-Probleme mit Befehlen in Programmen, die in anderen Tasks laufen, zu vermeiden.

Argumente
- **Befehl**
  Die Auswahl der durchzuführenden Host-Link-Operation. Siehe die erste Tabelle unten.
- **Schnittstelle**
  Die spezifizierte serielle Schnittstelle. 1 = serielle RS-232C-Schnittstelle 1; 2 = serielle RS-422A-Schnittstelle 2.
- **teilnehmer** (für **HLM_MREAD**, **HLM_TEST**, **HLM_ABORT** und **HLM_STWR**)
  Die Slave-Teilnehmernummer, an die der Host-Link-Befehl gesendet werden soll. Bereich: [0, 31].
- **betriebsart** (für **HLM_STWR**)
  Die Betriebsart der CPU-Baugruppe. 0 = PROGRAM-Betriebsart; 2 = MONITOR-Betriebsart; 3 = RUN-Betriebsart.
- **mc_bereich** (für **HLM_MREAD**)
  Der Speicherbereich des TJ1-MC__, aus dem Daten gelesen und zu dem Daten gesendet werden sollen. Siehe zweite Tabelle unten.
- **mc_offset** (für **HLM_MREAD**)
  Die Startadresse des TJ1-MC__-Speicherbereichs, aus dem gelesen werden soll. Bereich für VR-Variablen: [0, 1023]. Bereich für TABLE-Variablen: [0, 63999].

Beispiel **HLM_COMMAND(HLM_MREAD,1,12,MC_VR,233)**
Dieser Befehl liest den Modell-Code der CPU-Baugruppe des Host-Link-Slaves mit der Teilnehmeradresse 12 aus, der an die RS-232C-Schnittstelle angeschlossen ist. Das Ergebnis wird nach VR(233) geschrieben.
Wenn der angeschlossene Slave ein C200HX-PC ist, enthält VR(233) nach erfolgreicher Ausführung den Wert 12 (hex).

Beispiel **HLM_COMMAND(HLM_TEST,2,23)**
PRINT HLM_STATUS PORT(2)
Dieser Befehl überprüft die Host-Link-Kommunikation mit dem Host-Link-Slave (Teilnehmer 23), der an die RS-422A-Schnittstelle angeschlossen ist. Wenn der Parameter **HLM_STATUS** den Wert 0 enthält, ist die Kommunikation hergestellt.

Beispiel **HLM_COMMAND(HLM_INIT,2)**
HLM_COMMAND(HLM_ABORT,2,4)
Mit diesen zwei Befehlen werden die Host-Link-Operationen **INITIALIZE** und **ABORT** an der RS-422A-Schnittstelle 2 durchgeführt. Der Slave hat die Teilnehmernummer 4.

Beispiel **HLM_COMMAND(HLM_STWR,2,0,2)**
Wenn Daten über Host-Link an einen PC geschrieben werden, darf die CPU-Baugruppe nicht in der RUN-Betriebsart sein. Der Befehl **HLM_COMMAND** kann dazu verwendet werden, sie in die MONITOR-Betriebsart zu setzen. Der Slave hat die Teilnehmeradresse 0 und ist an die RS-232C-Schnittstelle angeschlossen.

Siehe auch **HLM_READ**, **HLM_COMMAND**, **HLM_STATUS**, **HLM_TIMEOUT**, **HLS_NODE**, **HLM_WRITE**, **SETCOM**.

Befehlswert	Beschreibung
HLM_MREAD (oder Wert 0)	Hierdurch wird der Host-Link-PC-Befehl **MODEL READ** (**MM**) zum Auslesen des Modell-Codes der CPU-Baugruppe durchgeführt. Das Ergebnis wird zu der TJ1-MC__-Variablen geschrieben, die durch **mc_bereich** und **mc_offset** spezifiziert ist.
HLM_TEST (oder Wert 1)	Hierdurch wird der Host-Link-Befehl **TEST** (**TS**) zur Überprüfung der richtigen Kommunikation durchgeführt, indem die Zeichenkette „MCW151 TEST STRING“ gesendet und die Echo-Zeichenkette überprüft wird. Überprüfen Sie das Ergebnis im Parameter **HLM_STATUS**.
HLM_ABORT (oder Wert 2)	Hierdurch wird der Host-Link-Befehl **ABORT** (**XZ**) zum Abbruch des Host-Link-Befehls durchgeführt, der aktuell verarbeitet wird. Bei dem Befehl **ABORT** wird keine Antwort empfangen.
HLM_INIT (oder Wert 3)	Hierdurch wird der Host-Link-Befehl **INITIALIZE** (******) zur Initialisierung des Übertragungvorgangs aller Slave-Baugruppen durchgeführt.
HLM_STWR (oder Wert 4)	Hierdurch wird der Host-Link-PC-Befehl **STATUS WRITE** (**MM**) zum Ändern der Betriebsart der CPU-Baugruppe durchgeführt.

mc_Bereichswert	Datenbereich
MC_TABLE (oder Wert 8)	TABLE-Variablen-Array
MC_VR (oder Wert 9)	Globales (VR-) Variablen-Array

3.2.140 HLM_READ

Typ Kommunikationsbefehl

Syntax **HLM_READ(schnittstelle, teilnehmer, pc_bereich, pc_offset, laenge, mc_bereich, mc_offset)**

Beschreibung Der Befehl **HLM_READ** liest Daten aus einem Host-Link-Slave aus, indem an die serielle Schnittstelle eine Host-Link-Befehlszeichenkette gesendet wird, die die Teilnehmernummer des Slaves enthält. Die empfangenen Antwortdaten werden entweder zu VR- oder TABLE-Variablen geschrieben. Jedes Datenwort wird an eine Variable übermittelt. Die maximale Datenlänge ist 30 Worte (Einzelrahmen-Übertragung).
Die Programmausführung wird angehalten, bis die Antwort-Zeichenkette empfangen wurde oder eine Zeitüberschreitung eingetreten ist. Die Zeitüberschreitung wird durch den Parameter **HLM_TIMEOUT** spezifiziert. Der Status der Übertragung kann mit dem Parameter **HLM_STATUS** überwacht werden.
Hinweise:

- Wenn Sie den Befehl **HLM_READ** benutzen, stellen Sie sicher, dass das Host-Link-Masterprotokoll eingerichtet ist, indem Sie den Befehl **SETCOM** verwenden.
- Die Host-Link-Masterbefehle dürfen nur von einem Programm ausgeführt werden, um Timing-Probleme mit Befehlen in Programmen, die in anderen Tasks laufen, zu vermeiden.

Argumente

- **Schnittstelle**
  Die serielle Schnittstelle. 1 = serielle RS-232C-Schnittstelle 1; 2 = serielle RS-422A-Schnittstelle 2.
- **Teilnehmer**
  Die Slave-Teilnehmernummer, an die der Host-Link-Befehl gesendet werden soll. Bereich: [0, 31].
- **pc_bereich**
  Die PC-Speicherauswahl für den Host-Link-Befehl. Siehe erste Tabelle unten.
- **pc_offset**
  Die Adresse des PC-Speicherbereichs, aus dem gelesen werden soll. Bereich: [0, 9999].
- **laenge**
  Die Anzahl der zu übertragenden Datenworte. Bereich: [1, 30].
- **mc_bereich**
  Der Speicherbereich des TJ1-MC__, aus dem Daten gelesen und zu dem Daten gesendet werden sollen. Siehe zweite Tabelle unten.
- **mc_Offset**
  Die Adresse des TJ1-MC__-Speicherbereichs, in den geschrieben werden soll. Bereich für VR-Variablen: [0, 1023]. Bereich für TABLE-Variablen: [0, 63999].

Beispiel **HLM_READ(2,17,PLC_DM,120,20,MC_TABLE,4000)**
Dieses Beispiel zeigt, wie 20 Worte aus den PC-DM-Bereichsadressen 120–139 an die TABLE-Adressen 4000–4019 des TJ1-MC__ ausgelesen werden. Der PC hat Slave-Teilnehmeradresse 17 und ist an die RS-422A-Schnittstelle angeschlossen.

Siehe auch **HLM_COMMAND**, **HLM_STATUS**, **HLM_TIMEOUT**, **HLS_NODE**, **HLM_WRITE**, **SETCOM**.

pc_bereichswert	Datenbereich	Host-Link-Befehl
PLC_DM (oder Wert 0)	DM-Bereich	RD
PLC_IR (oder Wert 1)	CIO/IR-Bereich	RR
PLC_LR (oder Wert 2)	LR-Bereich	RL
PLC_HR (oder Wert 3)	HR-Bereich	RH
PLC_AR (oder Wert 4)	AR-Bereich	RJ
PLC_EM (oder Wert 6)	EM-Bereich	RE

mc_Bereichswert	Datenbereich
MC_TABLE (oder Wert 8)	TABLE-Variablen-Array
MC_VR (oder Wert 9)	Globales (VR-) Variablen-Array

3.2.141 HLM_STATUS

Typ Kommunikationsparameter

Syntax **HLM_STATUS PORT(n)**

Beschreibung Der Parameter **HLM_STATUS** enthält den Status des letzten Host-Link-Masterbefehls, der an die im Befehl spezifizierte Schnittstelle gesendet wurde. Der Parameter zeigt den Status für die Befehle **HLM_READ**, **HLM_WRITE** und **HLM_COMMAND** an. Die Status-Bits sind in der nachstehenden Tabelle definiert.
Der Parameter **HLM_STATUS** hat den Wert 0, wenn keine Probleme aufgetreten sind. Ist der Wert nicht 0, muss im Anwender-BASIC-Programm eine geeignete Maßnahme, z. B. eine Befehlswiederholung oder ein Nothalt programmiert werden.
Jede Schnittstelle hat einen **HLM_STATUS**-Parameter. Der Modifikator **PORT** ist erforderlich, um die Schnittstelle zu spezifizieren.

Argumente

- **n**
  Die spezifizierte serielle Schnittstelle. 1 = serielle RS-232C-Schnittstelle 1; 2 = serielle RS-422A-Schnittstelle 2.

Beispiel
```
>> HLM_WRITE(1,28,PLC_EM,50,25,MC_VR,200)
>> PRINT HEX(HLM_STATUS PORT(1))
1
```
Anscheinend befindet sich die CPU-Baugruppe in der RUN-Betriebsart und akzeptiert den Schreibvorgang nicht.

Beispiel **>> HLM_COMMAND(HLM_TEST,2,0)**
>> PRINT HLM_STATUS PORT(2)
256.0000
Es ist ein Zeitüberschreitungsfehler aufgetreten.

Siehe auch **HLM_READ**, **HLM_COMMAND**, **HLM_TIMEOUT**, **HLS_NODE**, **HLM_WRITE**, **SETCOM**.

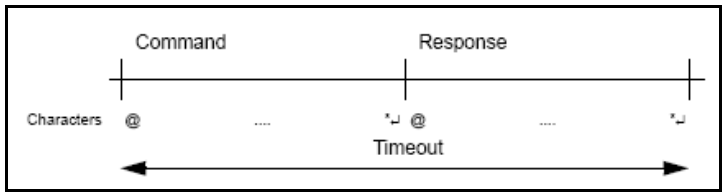
Bit	Name	Beschreibung
0–7	Endcode	Der Endcode kann entweder der Endcode sein, der durch den Host-Link-Slave definiert ist (Problem in der gesendeten Befehls-Zeichenkette) oder ein Endcode, der wegen eines Problems definiert wurde, das beim Host-Link-Master aufgetreten ist (Problem in der empfangenen Antwort-Zeichenkette).
8	Zeitüberschrei-tungsfehler	Ein Zeitüberschreitungsfehler tritt auf, wenn innerhalb der für die Zeitüberschreitung festgelegten Zeit keine Antwort empfangen wird. Dies deutet daraufhin, dass die Kommunikation unterbrochen ist.
9	Befehl nicht erkannt	Dieser Zustand deutet darauf hin, dass der Slave den Befehl nicht erkannt und eine IC-Antwort zurückgegeben hat.

3.2.142 HLM_TIMEOUT

Typ Kommunikationsparameter

Syntax **HLM_TIMEOUT**

Beschreibung Der Parameter **HLM_TIMEOUT** spezifiziert die für die Zeitüberschreitung festgelegte Zeit für das Host-Link-Masterprotokoll für beide serielle Schnittstellen. Ein Zeitüberschreitungsfehler tritt auf, wenn die Zeit, die zum Senden des Befehls und zum Empfangen der Antwort vom Slave benötigt wird, länger als die mit diesem Parameter spezifizierte Zeit ist. Der Parameter gilt für die Befehle **HLM_READ**, **HLM_WRITE** und **HLM_COMMAND**. Der Parameter **HLM_TIMEOUT** ist in Servozyklen spezifiziert.



Argumente –

Beispiel **>> HLM_TIMEOUT = 2000**
Beachten Sie, dass der Servozyklus des TJ1-MC__ auf 500 ms eingestellt ist (SERVO_PERIOD = 500). Für beide serielle Schnittstellen ist die Host-Link-Master-Zeitüberschreitung auf 1 s eingestellt.

Siehe auch **HLM_READ**, **HLM_COMMAND**, **HLM_STATUS**, **HLS_NODE**, **HLM_WRITE**, **SETCOM SERVO_PERIOD**.

3.2.143 HLM_WRITE

Typ Kommunikationsbefehl

Syntax **HLM_WRITE(schnittstelle, teilnehmer, pc_bereich, pc_offset, laenge, mc_bereich, mc_offset)**

Beschreibung Der Befehl **HLM_WRITE** schreibt Daten aus dem TJ1-MC__ zu einem Host-Link-Slave, indem an die serielle Schnittstelle eine Host-Link-Befehlszeichenkette gesendet wird, die die Teilnehmernummer des Slaves enthält. Die empfangenen Antwortdaten werden entweder aus VR- oder TABLE-Variablen geschrieben. Jede Variable definiert das Wort oder die Daten, die übermittelt werden. Die maximale Datenlänge ist 29 Worte (Einzelrahmen-Übertragung).
Die Programmausführung wird angehalten, bis die Antwort-Zeichenkette empfangen wurde oder eine Zeitüberschreitung eingetreten ist. Die Zeitüberschreitung wird durch den Parameter **HLM_TIMEOUT** spezifiziert. Der Status der Übertragung kann mit dem Parameter **HLM_STATUS** überwacht werden.
Hinweise:

- Wenn Sie den Befehl **HLM_WRITE** benutzen, stellen Sie sicher, dass das Host-Link-Masterprotokoll eingerichtet ist, indem Sie den Befehl **SETCOM** verwenden.
- Die Host-Link-Masterbefehle dürfen nur von einem Programm ausgeführt werden, um Timing-Probleme mit Befehlen in Programmen, die in anderen Tasks laufen, zu vermeiden.

Argumente

- **Schnittstelle**
  Die spezifizierte serielle Schnittstelle. 1 = serielle RS-232C-Schnittstelle 1; 2 = serielle RS-422A-Schnittstelle 2.
- **Teilnehmer**
  Die Slave-Teilnehmernummer, an die der Host-Link-Befehl gesendet werden soll. Bereich: [0, 31].
- **pc_bereich**
  Die PC-Speicherauswahl für den Host-Link-Befehl. Siehe erste Tabelle unten.
- **pc_offset**
  Die Adresse des PC-Speicherbereichs, in den geschrieben werden soll. Bereich: [0, 9999].
- **laenge**
  Die Anzahl der zu übertragenden Datenworte. Bereich: [1, 29].
- **mc_bereich**
  Der Speicherbereich des TJ1-MC__, aus dem Daten gelesen und zu dem Daten gesendet werden sollen. Siehe zweite Tabelle unten.
- **mc_Offset**
  Die Adresse des TJ1-MC__-Speicherbereichs, aus dem gelesen werden soll. Bereich für VR-Variablen: [0, 1023]. Bereich für TABLE-Variablen: [0, 63999].

Beispiel **HLM_WRITE(1,28,PLC_EM,50,25,MC_VR,200)**
Dieses Beispiel zeigt, wie 25 Worte aus den VR-Adressen 200–224 des TJ1-MC__ an die EM-Bereichsadressen 50–74 des PCs geschrieben werden. Der PC hat Slave-Teilnehmeradresse 28 und ist an die RS-232C-Schnittstelle angeschlossen.

Siehe auch **HLM_READ**, **HLM_COMMAND**, **HLM_STATUS**, **HLM_TIMEOUT**, **HLS_NODE**, **SETCOM**.

pc_bereichswert	Datenbereich	Host-Link-Befehl
PLC_DM (oder Wert 0)	DM-Bereich	RD
PLC_IR (oder Wert 1)	CIO/IR-Bereich	RR
PLC_LR (oder Wert 2)	LR-Bereich	RL

pc_bereichswert	Datenbereich	Host-Link-Befehl
PLC_HR (oder Wert 3)	HR-Bereich	RH
PLC_AR (oder Wert 4)	AR-Bereich	RJ
PLC_EM (oder Wert 6)	EM-Bereich	RE

mc_Bereichswert	Datenbereich
MC_TABLE (oder Wert 8)	TABLE-Variablen-Array
MC_VR (oder Wert 9)	Globales (VR-) Variablen-Array

3.2.144 HLS_NODE

Typ Kommunikationsparameter

Syntax **HLS_NODE**

Beschreibung Der Parameter **HLS_NODE** definiert die Slave-Baugruppennummer für das Host-Link-Slave-Protokoll. Der TJ1-MC__ antwortet auf die Host-Link-Master-Befehlszeichenketten nur mit der in diesem Parameter spezifizierten Baugruppennummer. Der gültige Bereich für diesen Parameter ist [0, 31]. Der Standardwert ist 0.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.145 HW_PSWITCH

Typ Achsenbefehl

Syntax **HW_PSWITCH(betriebsart, richtung, ausstatus, tabellen_anfang, tabellen_ende)**

Beschreibung Der Befehl **HW_PSWITCH** schaltet OUT 0 für die Achse ein, wenn die Einschaltposition für die Achse erreicht ist. Ausgang OUT 0 wird ausgeschaltet, wenn die Ausschaltposition erreicht ist. Positionen sind im TABLE-Speicher als Wertefolge im Bereich von **Tabellen_anfang** bis **Tabellen_ende** definiert und werden bei Ausführung des Befehls **HW_PSWITCH** in der FIFO-Warteschlange gespeichert.
Der Parameter ist nur auf Flex Achsmodule mit den**ATYPE**-Werten 43, 44 und 45 anwendbar.
Der Befehl kann zusammen mit einem oder mit 5 Parametern verwendet werden. Es ist nur ein Parameter erforderlich, um den Befehl zu deaktivieren oder die FIFO-Warteschlange zu löschen. Alle fünf Parameter sind erforderlich, um den Befehl zu aktivieren.
Nach dem Laden der FIFO-Warteschlange und dem Abarbeiten der Positionsfolge in der FIFO-Schlange muss diese gelöscht werden. Erst danach kann der Befehl **HW_PSWITCH** mit den gleichen Parametern erneut ausgeführt werden.

Argumente
- **Modus**
  0 = Schalter deaktivieren; 1 = EIN und FIFO laden; 2 = FIFO löschen.
- **Richtung**
  0 = absteigend; 1 = aufsteigend.
- **Ausstatus**
  Ausgangsstatus, der in der ersten Position in der FIFO eingestellt ist; EIN oder AUS.
- **Tabellen_anfang**
  Tabellen-Anfangsadresse der Folge.
- **Tabellen_ende**
  Tabellen-Endadresse der Folge.

Beispiel **HW_PSWITCH(1, 1, ON, 21, 50)**
Dieser Befehl lädt die FIFO mit 30 im TABLE-Speicher gespeicherten Positionen in aufsteigender Richtung ab **TABLE(21)**. Wenn die in **TABLE(21)** gespeicherte Position erreicht ist, wird Ausgang OUT 0 eingeschaltet und anschließend abwechselnd beim Erreichen der folgenden Positionen in der Folge aus- und eingeschaltet, bis die in **TABLE(50)** gespeicherte Position erreicht ist.

Beispiel **HW_PSWITCH(0)**
Dieser Befehl schaltet den Schalter aus, wenn er zuvor eingeschaltet war, löscht jedoch nicht die FIFO-Warteschlange.

Beispiel **HW_PSWITCH(2)**
Dieser Befehl löscht die FIFO-Warteschlange, wenn sie vorher geladen war.

Siehe auch **AXIS**

3.2.146 I_GAIN

Typ Achsenparameter

Syntax **I_GAIN**

Beschreibung Der Achsenparameter **I_GAIN** enthält die Integralverstärkung für die Achse. Die integrale Leistungsbeeinflussung wird berechnet, indem die Summen der Schleppfehler mit dem Wert des Parameters **I_GAIN** multipliziert werden. Der Standardwert ist 0.
Das Hinzufügen einer Integralverstärkung zu einem Servosystem verringert den Positionierfehler im Ruhezustand oder bei stetigen Bewegungen.
Die Integralverstärkung kann Überschwingen und Schwingungen verursachen und ist deshalb nur für Systeme geeignet, die mit konstanter Drehzahl oder langsamer Beschleunigung arbeiten.
Hinweis: Um Instabilität zu vermeiden, sollte die Servoverstärkung nur dann geändert werden, wenn **SERVO** ausgeschaltet ist.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **D_GAIN**, **I_GAIN**, **OV_GAIN**, **P_GAIN**, **VFF_GAIN**.

3.2.147 IDLE

Siehe **WAIT IDLE**.

3.2.148 IEEE_IN

Typ Mathematische Funktion

Syntax **IEEE_IN(byte0,byte1,byte2,byte3)**

Beschreibung Die Funktion **IEEE_IN** gibt die durch vier Byte dargestellte Gleitkommazahl zurück, die normalerweise über eine Kommunikationsverbindung empfangen wurde.

Argumente
- **Byte0** – Byte3
  Jede Kombination von 8-Bit-Werten, die eine gültige IEEE-Gleitkommazahl darstellt.

Beispiel **VR(20) = IEEE_IN(b0,b1,b2,b3)**

Siehe auch –

3.2.149 IEEE_OUT

Typ Mathematische Funktion

Syntax **Byte_n = IEEE_OUT(wert, n)**

Beschreibung Die Funktion **IEEE_OUT** gibt ein einzelnes Byte im IEEE-Format zurück, das aus dem Gleitkommawert zur Übertragung über ein Bussystem extrahiert wurde. Die Funktion wird normalerweise viermal aufgerufen, um jedes Byte nacheinander zu extrahieren.
Hinweis: Byte 0 ist die höherwertige Bytezahl des 32-Bit-IEEE-Gleitkommaformats.

Argumente
- **Wert**
  Jede(r) BASIC-Gleitkommavariable oder -parameter.
- **n**
  Die Nummer des zu extrahierenden Bytes (0 bis 3).

Beispiel
```
V = MPOS AXIS(2)
Byte0 = IEEE_OUT(V, 0)
Byte1 = IEEE_OUT(V, 1)
Byte2 = IEEE_OUT(V, 2)
Byte3 = IEEE_OUT(V, 3)
```

Siehe auch –

3.2.150 IF..THEN..ELSE..ENDIF

Typ Programmsteuerungsbefehl

Syntax **IF bedingung_1 THEN Befehle {ELSEIF bedingung_i THEN Befehle}**
[ELSE Befehle] ENDIF
IF bedingung_1 THEN Befehle

Beschreibung Diese Struktur steuert den Programmablauf basierend auf den Ergebnissen der Bedingung. Wenn die Bedingung **TRUE** ist, werden die Befehle ausgeführt, die auf **THEN** bis **ELSEIF**, **ELSE** oder **ENDIF** folgen. Wenn die Bedingung **FALSE** ist und der Befehl einer darauf folgenden **ELSEIF**-Unterstruktur **TRUE** ist, werden die Befehle dieser Unterstruktur ausgeführt. Wenn alle Bedingungen **FALSE** sind, werden die auf **ELSE** folgenden Befehle ausgeführt, oder das Programm nimmt die Zeile nach **ENDIF** wieder auf, falls keine **ELSE**-Bedingung enthalten ist. **ENDIF** wird verwendet, um das Ende des Bedingungsblocks zu kennzeichnen.
Hinweis: **IF..THEN..ELSE..ENDIF**-Folgen können unbegrenzt verschachtelt werden. Bei einer mehrzeiligen **IF..THEN**-Konstruktion darf keine Anweisung hinter **THEN** stehen. Bei einer einzeiligen Konstruktion darf kein **ENDIF** verwendet werden.

Argumente
- **Bedingung_i**
  Ein logischer Ausdruck.
- **Befehle**
  Einer oder mehrere BASIC-Befehle.

Beispiel
```
IF MPOS > (0,22 * VR(0)) THEN GOTO position_gross
```

Beispiel
```
IF IN(0) = ON THEN
 nummer = nummer + 1
 PRINT "Nummer = ";nummer
 fehlschlag = 0
ELSE
 fehlschlag = fehlschlag + 1
ENDIF
```

Beispiel
```
IF IN(stopp) = ON THEN
 OP(8,ON)
 VR(zyklus_merker) = 0
ELSEIF IN(start_zyklus) = ON THEN
 VR(zyklus_merker) = 1
ELSEIF IN(schritt1) = ON THEN
 VR(zyklus_merker) = 99
ENDIF
```

Beispiel
```
IF buchstabe_zeich = $31 THEN
 GOSUB zeich_1
ELSEIF buchstabe_zeich = $32 THEN
 GOSUB zeich_2
ELSEIF buchstabe_zeich = $33 THEN
 GOSUB zeich_3
ELSE
 PRINT "Zeichen unbekannt"
ENDIF
```

Siehe auch –

3.2.151 IN

Typ E/A-Funktion

Syntax **IN(eingangs_nummer [, letzte_eingangs_nummer])**
IN

Beschreibung Die Funktion **IN** gibt den Wert digitaler Eingänge zurück.
- **IN (eingangs_nummer, letzte_eingangs_nummer)** gibt die Binärsumme der Eingangsgruppe zurück. Die beiden Argumente dürfen nicht mehr als 24 auseinander liegen.
- **IN (eingangs_nummer)** mit einem Wert für **Eingangs_nummer**, der unter 32 liegt, gibt den Wert des entsprechenden Kanals zurück.
- **IN** (ohne Argumente) gibt die Binärsumme der ersten 24 Eingänge zurück (als **IN(0,23)**).

Argumente
- **eingangs_nummer**
  Die Nummer des Eingangs, für den ein Wert zurückgegeben werden soll. Wert: Eine Integer-Zahl zwischen 0 und 31.
- **letzte_eingangs_nummer**
  Die Nummer des letzten Eingangs, für den ein Wert zurückgegeben werden soll. Wert: Eine Integer-Zahl zwischen 0 und 31.

Beispiel

Die folgenden Zeilen können dazu verwendet werden, die auf einem Daumenradschalter eingestellte Position zu bewegen, die mit einem Faktor multipliziert wird. Der Daumenradschalter ist mit den Eingängen 4, 5, 6 und 7 verbunden und gibt Werte in BCD aus.

```
bewegschleife:
MOVEABS(IN(4,7) * 1.5467)
WAIT IDLE
GOTO bewegschleife
```

Der Befehl **MOVEABS** ist wie folgt aufgebaut:
Schritt 1: **IN(4,7)** ergibt eine Zahl zwischen 0 und 15.
Schritt 2: Die Zahl wird mit 1,5467 multipliziert, um die benötigten Entfernung zu erhalten.
Schritt 3: Es wird eine absolute Bewegung zu dieser Position ausgeführt.

Beispiel

In diesem Beispiel wird ein einzelner Eingang getestet:

```
Test:
  WAIT UNTIL IN(4) = ON ' Band ist in Position wenn ON
  GOSUB stelle
```

Siehe auch **OP**.

3.2.152 INDEVICE

Typ E/A-Parameter

Syntax **INDEVICE**

Beschreibung Der Parameter **INDEVICE** definiert das Standard-Eingabegerät. Dieses Gerät wird für die Eingabebefehle ausgewählt, wenn die Option **#n** ausgelassen wird. Der Parameter **INDEVICE** ist Task-spezifisch. Die unterstützten Werte sind in der nachstehenden Tabelle aufgeführt.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **GETGET**, **INPUT**, **LINPUT**, **KEY**.

Wert	Beschreibung
0	Programmierschnittstelle 0 (standard)
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.153 INITIALISE

Typ Systembefehl

Syntax **INITIALISE**

Beschreibung Setzt alle Achsen-, System- und Prozessparameter auf ihre Standardwerte zurück. Die Parameter werden auch jedes Mal zurückgesetzt, wenn der Controller hochgefahren wird oder wenn ein **EX**- (Software-Rücksetz-) Befehl ausgeführt wird. In Trajexia Tools entspricht der Menüpunkt **Controller zurücksetzen...** im Menü **Controller** der Funktion eines **EX**-Befehls.

Argumente –

Beispiel Kein Beispiel.

Siehe auch
- **EX**

3.2.154 INPUT

Typ E/A-Befehl

Syntax **INPUT [#n], Variable { , Variable }**

Beschreibung Der Befehl **INPUT** ordnet den spezifizierten Variablen numerische Eingabe-Zeichenkettenwerte zu. Mehrere Eingabe-Zeichenkettenwerte können in einer Zeile, durch Kommas getrennt oder in mehreren Zeilen, die durch Wagenrücklauf getrennt sind, angefordert werden. Die Programmausführung wird angehalten, bis die Zeichenkette mit einem Wagenrücklauf beendet ist, nachdem die letzte Variable zugeordnet wurde.
Wenn die Zeichenkette ungültig ist, wird eine Fehlermeldung angezeigt, und der Task wird wiederholt. Die maximale Anzahl von Eingaben in einer Zeile ist nur durch die Zeilenlänge begrenzt.
Die Kanäle 5 bis 7 sind logische Kanäle, die bei der Verwendung von Trajexia Tools der RS-232C-Programmierschnittstelle 0 übergeordnet sind.
Hinweis: Kanal 0 ist für die Verbindung zu Trajexia Tools und/oder der Befehlszeilenschnittstelle reserviert. Bitte achten Sie darauf, dass es bei diesem Kanal für diese Funktion Probleme geben kann.

Argumente
- **n**
  Das spezifizierte Eingabegerät. Wird dieses Argument ausgelassen, wird die Schnittstelle verwendet, die durch INDEVICE spezifiziert ist.
- **Variable**
  Die Variable, zu der geschrieben werden soll.

Beispiel Das folgende Programm dient dazu, Daten vom Terminal zu empfangen.
INPUT#5, Num
PRINT#5, "BATCH COUNT = ";num[0]
Eine mögliche Antwort vom Terminal könnte sein:
123<CR>
BATCH COUNT = 123

Siehe auch **INDEVICE**, **GET**, **LINPUT**, **KEY**

3.2.155 INT

Typ Mathematische Funktion

Syntax **INT(ausdruck)**

Beschreibung Die Funktion **INT** gibt den Integer-Teil des Ausdrucks zurück.
Hinweis: Um eine positive Zahl auf den nächsten Integer-Wert zu runden, nehmen Sie die **INT**-Funktion des Wertes plus 0,5. Ziehen Sie zum Runden bei einem negativen Wert auf ähnliche Weise 0,5 von dem Wert ab, bevor Sie **INT** anwenden.

Argumente
- **Ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel **>> PRINT INT(1.79)**
1.0000

Siehe auch –

3.2.156 INVERT_IN

Typ Systembefehl

Syntax **INVERT_IN(eingang,ein/aus)**

Beschreibung Mit dem Befehl **INVERT_IN** können die Eingangskanäle 0..31 softwareseitig einzeln invertiert werden.
Dies ist wichtig, da diese Eingangskanäle aktiven Funktionen zugeordnet werden können, z. B. Feedhold.
Die Funktion **INVERT_IN** schaltet die Invertierung für einen Kanal ein oder aus. Sie kann nur auf die Eingänge 0..31 angewendet werden.

Argumente
- **Eingang**
  Ein BASIC-Ausdruck.

Beispiel **>>? IN(3)**
0.0000
>>INVERT_IN(3,ON)
>>? IN(3)
1.0000

Siehe auch –

3.2.157 INVERT_STEP

Typ	Achsenparameter
Syntax	**INVERT_STEP**
Beschreibung	**INVERT_STEP** dient dazu, eine Hardware-Invertierung in den Schrittmotor-Ausgangsschaltkreis einzufügen. Dies kann für den Anschluß einiger, Schrittmotor-Treiber notwendig sein. Die elektronische Logik im Trajexia-Schrittmotor-Impulsgeber geht davon aus, dass die FALLENDE Flanke des Schrittausgangs die aktive Flanke ist, die eine Motorbewegung zur Folge hat. Dies ist für die meisten Schrittmotor-Treiber geeignet. Die Einstellung **INVERT_STEP = ON** macht die STEIGENDE Flanke des Schrittsignals zur aktiven Flanke. **Falls erforderlich,** sollte INVERT_STEP eingestellt werden, bevor der Controller mit **WDOG = ON** aktiviert wird. Die Standardeinstellung ist AUS. Hinweis: Falls die Einstellung falsch ist, kann ein Schrittmotor die Position um einen Schritt verlieren, wenn die Richtung geändert wird.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.158 INVERTER_COMMAND

Typ	Systembefehl
Syntax	**INVERTER_COMMAND(modul, station, 7, betriebs_signale)** **INVERTER_COMMAND(modul, station, 1, alarm_nummer)**
Beschreibung	Der Befehl **INVERTER_COMMAND** steuert Eingänge und löscht Alarme eines Frequenzumrichters, der über den MECHATROLINK-II-Bus an das System angeschlossen ist. Es gibt zwei **INVERTER_COMMAND**-Funktionen: • 1: Löscht einen Alarm. • 7: Steuert Betriebssignale. Um einen Frequenzumrichter über MECHATROLINK-II zu verwenden, sollten Sie den Befehl und den Sollwert über folgende Kommunikationsoption eingeben: • Frequenzumrichter MV/V7: N3 = 3; N4 = 9 • Frequenzumrichter F7/G7: B1-01 = 3; B1-02 = 3. Vergewissern Sie sich, dass die Frequenzumrichter-Firmware die MECHATROLINK-II-Karte unterstützt. Bei erfolgreicher Ausführung gibt der Befehl –1, andernfalls 0 zurück. Der an den Frequenzumrichter gesendete Befehl entspricht den in der Tabelle unten angegebenen Bits.
Argumente	• **Modul** Die Anzahl der TJ1-ML__, an die der Frequenzumrichter angeschlossen ist. • **Station** Die MECHATROLINK-II-Stationsnummer des Frequenzumrichters. • **Alarm_nummer** Die Nummer des Alarms. Weitere Informationen finden Sie im Frequenzumrichter-Handbuch. • **Betriebs_signale** Ein in Bits unterteilter Wert zur Steuerung der Betriebssignale. Siehe Tabelle unten.
Beispiel	Kein Beispiel.
Siehe auch	–

Bit	Wert	Befehl	Beschreibung
0	Hex	1	Vorwärtslauf
1	Hex	2	Rückwärtslauf
2	Hex	4	Multifunktionseingang 3 des Frequenzumrichters
3	Hex	8	Multifunktionseingang 4 des Frequenzumrichters
4	Hex	10	Multifunktionseingang 5 des Frequenzumrichters
5	Hex	20	Multifunktionseingang 6 des Frequenzumrichters
6	Hex	40	Multifunktionseingang 7 des Frequenzumrichters
7	Hex	80	Multifunktionseingang 8 des Frequenzumrichters (nur G7)
8	Hex	100	Externer Fehler
9	Hex	200	Fehlerrücksetzung
10	Hex	400	Multifunktionseingang 9 des Frequenzumrichters (nur G7)
11	Hex	800	Multifunktionseingang 10 des Frequenzumrichters (nur G7)
12	Hex	1000	Multifunktionseingang 11 des Frequenzumrichters (nur G7)
13	Hex	2000	Multifunktionseingang 12 des Frequenzumrichters (nur G7)
14	Hex	4000	Löschen von Fehlerhistoriendaten
15	Hex	8000	Externer BB-Befehl

3.2.159 INVERTER_READ

Typ Systembefehl

Syntax **INVERTER_READ(modul, station, 0, param_nummer, param_groesse, VR)**
INVERTER_READ(modul, station, 1, alarm_nummer, VR)
INVERTER_READ(modul, station, 2, VR)
INVERTER_READ(modul, station, 3, VR)
INVERTER_READ(modul, station, 4, ab, laenge, VR)

Beschreibung Der Befehl **INVERTER_READ** liest den Parameter, den Drehzahlsollwert, Drehmomentsollwert oder Alarm des Frequenzumrichters aus, der über den MECHATROLINK-II-Bus an das System angeschlossen ist.
Es gibt fünf **INVERTER_READ**-Funktionen:

- 0: Liest einen Frequenzumrichter-Parameter.
- 1: Liest den Frequenzumrichter-Alarm.
- 2: Liest den Drehzahlsollwert.
- 3: Liest den Drehmomentsollwert.
- 4: Liest die Frequenzumrichter-Eingänge.

Um einen Frequenzumrichter über MECHATROLINK-II zu verwenden, sollten Sie den Befehl und den Sollwert über folgende Kommunikationsoption eingeben:

- Frequenzumrichter MV/V7: N3 = 3; N4 = 9
- Frequenzumrichter F7/G7: B1-01 = 3; B1-02 = 3.

Vergewissern Sie sich, dass die Frequenzumrichter-Firmware die MECHATROLINK-II-Karte unterstützt.
Bei erfolgreicher Ausführung gibt der Befehl 1, andernfalls 0 zurück. Das Ergebnis wird (falls vorhanden) in der ausgewählten VR zurückgegeben.

Argumente
- **Modul**
  Die Anzahl der TJ1-ML__, an die der Frequenzumrichter angeschlossen ist.
- **Station**
  Die MECHATROLINK-II-Stationsnummer des Frequenzumrichters.
- **param_nummer**
  Die Nummer des Parameters, der gelesen werden soll.
  Weitere Informationen finden Sie im Frequenzumrichter-Handbuch.
- **param_groesse**
  Die Größe des Parameters, der gelesen werden soll, 2 oder 4 Byte.
  Weitere Informationen finden Sie im Frequenzumrichter-Handbuch.
- **VR**
  Die Adresse im VR-Speicher des TJ-MC__, an der die gelesenen Informationen abgelegt werden. Wenn die Funktion **4** ist, wird das Ergebnis als bitweiser Wert zurückgegeben. Siehe Tabelle unten.
- **alarm_nummer**
  Die Nummer des Alarms, der gelesen werden soll. Weitere Informationen finden Sie im Frequenzumrichter-Handbuch.
- **ab**
  Die Startadresse des zu lesenden Eingangs.
- **laenge**
  Die Länge des zu lesenden Eingangs.

Beispiel Kein Beispiel.

Siehe auch –

Bit	Wert	Befehl	Beschreibung
0	Hex	1	Vorwärtslauf
1	Hex	2	Rückwärtslauf
2	Hex	4	Multifunktionseingang 3 des Frequenzumrichters
3	Hex	8	Multifunktionseingang 4 des Frequenzumrichters
4	Hex	10	Multifunktionseingang 5 des Frequenzumrichters
5	Hex	20	Multifunktionseingang 6 des Frequenzumrichters
6	Hex	40	Multifunktionseingang 7 des Frequenzumrichters
8	Hex	100	Externer Fehler
9	Hex	200	Fehlerrücksetzung
14	Hex	4000	Löschen von Fehlerhistoriendaten
15	Hex	8000	Externer BB-Befehl

3.2.160 INVERTER_WRITE

Typ Systembefehl

Syntax **INVERTER_WRITE(modul, station, 0, param_nummer, param_groesse, VR, modus)**
INVERTER_WRITE(modul, station, 2, wert)
INVERTER_WRITE(modul, station, 3, wert)

Beschreibung Der Befehl **INVERTER_WRITE** schreibt den Parameter, den Drehzahlsollwert oder Drehmomentsollwert des Frequenzumrichters, der über den MECHATROLINK-II-Bus an das System angeschlossen ist.
Es gibt drei **INVERTER_WRITE**-Funktionen:
- 0: Schreibt einen Frequenzumrichter-Parameter.
- 2: Schreibt den Drehzahlsollwert.
- 3: Schreibt den Drehmomentsollwert.

Um einen Frequenzumrichter über MECHATROLINK-II zu verwenden, sollten Sie den Befehl und den Sollwert über folgende Kommunikationsoption eingeben:
- Frequenzumrichter MV/V7: N3 = 3; N4 = 9.
- Frequenzumrichter F7/G7: B1-01 = 3; B1-02 = 3.

Vergewissern Sie sich, dass die Frequenzumrichter-Firmware die MECHATROLINK-II-Karte unterstützt.
Bei erfolgreicher Ausführung gibt der Befehl –1, andernfalls 0 zurück.
Das Ergebnis wird (falls vorhanden) in der ausgewählten VR zurückgegeben.

Argumente	• **Modul** Die Anzahl der TJ1-ML__, an die der Frequenzumrichter angeschlossen ist. • **station** Die MECHATROLINK-II-Stationsnummer des Frequenzumrichters. • **param_nummer** Die Nummer des Parameters, der geschriebenen werden soll. Weitere Informationen finden Sie im Frequenzumrichter-Handbuch. • **param_groesse** Die Größe des Parameters, der geschrieben werden soll, 2 oder 4 Byte. Weitere Informationen finden Sie im Frequenzumrichter-Handbuch. • **VR** Die Adresse im VR-Speicher des TJ1-MC__, an der der neue Wert für den Parameter gespeichert ist. • **Modus** 0 = nur schreiben; 1 = schreiben und eingeben; 2 = schreiben und konfigurieren. • **Wert** Der neue Wert, der geschrieben wird.
Beispiel	Kein Beispiel.
Siehe auch	–

Wenn Sie viele Parameter gleichzeitig übertragen müssen, besteht die effizienteste Methode darin, MODE 0 für alle außer dem letzten Parameter und MODE 1 für den letzten Parameter zu verwenden. MODE 0 wird schneller ausgeführt als MODE 1.

3.2.161 JOGSPEED

Typ	Achsenparameter
Syntax	**JOGSPEED**
Beschreibung	Der Parameter **JOGSPEED** stellt die Jog-Drehzahl für eine Achse in Benutzereinheiten ein. Ein Tippschritt wird ausgeführt, wenn ein Tippeingang für eine Achse definiert wurde und dieser Eingang null ist. Ein Vorwärtstippeingang und ein Rückwärtstippeingang sind für jede Achse verfügbar bzw. werden durch die Parameter **FWD_JOG** und **REV_JOG** eingestellt. Die Tippgeschwindigkeit kann mit dem **FAST_JOG**-Eingang gesteuert werden.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **FAST_JOG**, **FWD_JOG**, **REV_JOG**, **UNITS**.

3.2.162 KEY

Typ	E/A-Parameter
Syntax	**KEY [#n]**
Beschreibung	Der Parameter **KEY** gibt **TRUE** oder **FALSE** zurück, je nachdem, ob ein Zeichen im Eingangsspeicher der seriellen Schnittstelle empfangen wurde oder nicht. Ein **TRUE**-Ergebnis wird zurückgesetzt, wenn das Zeichen mit dem Befehl **GET** gelesen wird. Die Kanäle 5 bis 7 sind logische Kanäle, die bei der Verwendung von Trajexia Tools der Programmierschnittstelle 0 übergeordnet sind. Hinweis: Kanal 0 ist für die Verbindung zu Trajexia Tools und/oder der Befehlszeilenschnittstelle reserviert. Bitte achten Sie darauf, dass es bei diesem Kanal für diese Funktion Probleme geben kann.
Argumente	• **n** Das spezifizierte Eingabegerät. Wird dieses Argument ausgelassen, wird die Schnittstelle verwendet, die durch **INDEVICE** spezifiziert ist. Siehe Tabelle unten.
Beispiel	**WAIT UNTIL KEY#1** **GET#1, k** Achten Sie darauf, dass bei der Verwendung von **KEY#1** in einer Gleichung in der Anweisung Klammern gesetzt werden müssen, in diesem Fall: **WAIT UNTIL (KEY#1) = TRUE**.

Siehe auch • **GET**

Wert	Eingabegerät
0	Programmierschnittstelle 0
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.163 LAST_AXIS

Typ Systemparameter

Syntax **LAST_AXIS** (Nur-Lesen)

Beschreibung Der Parameter **LAST_AXIS** enthält die Nummer der letzten Achse, die vom System bearbeitet wurde.
Die meisten Systeme verwenden nicht alle verfügbaren Achsen. Daher wäre es Zeitverschwendung, allen nicht vorhandenen Achsen Tasks zuzuweisen. Um dies bis zu einem gewissen Grad zu vermeiden, weist der TJ1-MC__ den Achsen von 0 bis **LAST_AXIS** Bewegungen zu, wobei **LAST_AXIS** die Nummer der höchsten Achse ist, für die ein **AXIS**- oder **BASE**-Befehl verarbeitet wurde, je nachdem, welcher von beiden größer ist.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **BASE**.

3.2.164 LINKAX

Typ Achsenparameter (Nur-Lesen)

Syntax **LINKAX**

Beschreibung Gibt die Nummer der Achse zurück, mit der eine Achse bei synchronisierten Bewegungen verknüpft ist. Synchronisierte Bewegungen sind dann gegeben, wenn die Sollposition eine Funktion einer anderen Achse ist.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **CONNECT**, **CAMBOX**, **MOVELINK**.

3.2.165 LINPUT

Typ E/A-Befehl

Syntax **LINPUT [#n ,] vr_Variable**

Beschreibung Der Befehl **LINPUT** ordnet den ASCII-Code der Zeichen einem Variablen-Array zu, das mit der spezifizierten VR-Variable beginnt. Die Programmausführung wird angehalten, bis die Eingabe der Zeichenkette mit einem Wagenrücklauf beendet ist. Die Zeichenkette wird mit dem Wagenrücklauf gespeichert.
Die Zeichenkette wird vom Controller nicht ausgegeben.
Die Kanäle 5 bis 7 sind logische Kanäle, die bei der Verwendung von Trajexia Tools der Programmierschnittstelle 0 übergeordnet sind.
Hinweis: Kanal 0 ist für die Verbindung zu Trajexia Tools und/oder der Befehlszeilenschnittstelle reserviert. Bitte achten Sie darauf, dass es bei diesem Kanal für diesen Befehl Probleme geben kann.

Argumente
- **n**
  Das spezifizierte Eingabegerät. Wird dieses Argument ausgelassen, wird die Schnittstelle verwendet, die durch INDEVICE spezifiziert ist. Siehe Tabelle unten.
- **vr_variable**
  Die erste VR-Variable, zu der geschrieben werden soll.

Beispiel Betrachten Sie die folgende Zeile in einem Programm.
LINPUT#5, VR(0)
Die Eingabe von START<CR> ergibt
VR(0) = 83S
VR(1) = 84T
VR(2) = 65A
VR(3) = 82R
VR(4) = 84T
VR(5) = 13<CR>

Siehe auch
- **GET**
- **INPUT**
- **VR**

Wert	Eingabegerät
0	Programmierschnittstelle 0
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.166 LIST

Typ Programmbefehl (nur Trajexia Tools-Befehlszeile)

Syntax **LIST ["programm_name"]**
TYPE ["programm_name"]

Beschreibung Nur zur Verwendung mit dem Terminalfenster. **LIST** wird nur als unmittelbarer (Befehlszeilen-) Befehl verwendet und darf nicht in Programmen benutzt werden. Der Befehl **LIST** druckt das aktuell ausgewählte Programm oder das Programm, das durch **Programm_name** spezifiziert ist. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. Wird der Programmname ausgelassen, wird das aktuell ausgewählte Programm aufgelistet.
Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann das Terminalfenster benutzt werden.

Argumente
- **programm_name**
  Das Programm, das gedruckt werden soll.

Beispiel Kein Beispiel.

Siehe auch **SELECT**.

3.2.167 LIST_GLOBAL

Typ Systembefehl (nur Terminal)

Syntax **LIST_GLOBAL**

Beschreibung Bei Ausführung über die Befehlszeile (Terminalkanal 0) gibt der Befehl alle aktuell eingestellten **GLOBAL**- und **CONSTANT**-Parameter zurück.

Argumente –

Beispiel In einer Anwendung, bei der die folgenden GLOBAL- und CONSTANT-Parameter eingestellt sind:
CONSTANT "schneidemaschine", 23
GLOBAL "foerderband",5

>>LIST_GLOBAL
Global VR
---------- ----
Förderband 5
Konstantenwert
---------- -------
Schneidemaschine 23.0000

Siehe auch –

3.2.168 LN

Typ Mathematische Funktion

Syntax **LN(ausdruck)**

Beschreibung Die Funktion **LN** gibt den natürlichen Logarithmus des Ausdrucks zurück. Der Wert des Eingangsausdrucks muss größer als 0 sein.

Argumente
- **Ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel **>> PRINT LN(10)**
2.3026

Siehe auch –

3.2.169 LOCK

Typ Systembefehl

Syntax **LOCK(code)**
UNLOCK(code)

Beschreibung Der Befehl **LOCK** verhindert, dass das Programm durch Personal, das den Sicherheitscode nicht kennt, angezeigt, modifiziert oder gelöscht wird. Der Befehl **UNLOCK** ermöglicht die Aufhebung des Sperrzustandes. Die Codenummer kann eine Integer-Zahl sein und wird in verschlüsselter Form gespeichert. **LOCK** ist immer ein unmittelbarer Befehl und kann nur eingegeben werden, wenn sich das System im Zustand **UNLOCKED** befindet.

Argumente
- **Code**
  Jede gültige Integer-Zahl mit maximal 7 Stellen.

Beispiel **>> LOCK(561234)**
Die Programme können nicht modifiziert oder angezeigt werden.
>> UNLOCK(561234)
Das System ist jetzt entsperrt.

Siehe auch –

Achtung
Der Sicherheitscode muss gut aufbewahrt werden; er ist erforderlich, um das System zu entsperren. Ohne den Sicherheitscode kann das System nicht wiederhergestellt werden.

3.2.170 MARK

Typ Achsenparameter (Nur-Lesen)

Syntax **MARK**

Beschreibung Der Parameter **MARK** wird auf **FALSE** gesetzt, wenn der Befehl **REGIST** ausgeführt wurde und auf **TRUE**, wenn die Druckmarke erkannt wurde.

Argumente –

Beispiel
```
IF MARK AXIS(1) THEN
  PRINT "Druckmarke fuer Achse 1 erkannt"
ENDIF
```

Siehe auch **AXIS**, **REGIST**, **REG_POS**.

3.2.171 MARKB

Typ Achsenparameter (Nur-Lesen)

Syntax **MARKB**

Beschreibung Der Parameter **MARKB** wird auf **FALSE** gesetzt, wenn der Befehl **REGIST** ausgeführt wurde und auf **TRUE**, wenn die zweite Druckmarke erkannt wurde. Dieser Parameter ist nur für FL02-Achsen verfügbar.

Argumente –

Beispiel
```
IF MARKB AXIS(2) THEN
  PRINT "Zweite Druckmarke fuer Achse 2 erkannt"
ENDIF
```

Siehe auch **AXIS**, **REGIST**, **REG_POSB**.

3.2.172 MECHATROLINK

Typ Systembefehl

Syntax **MECHATROLINK(baugruppe,0)**
Erkennt und verbindet Geräte über eine MECHATROLINK-II- **Baugruppe**. Der Befehl ist notwendig, um das Netzwerk bei einem Kommunikationsproblem zurückzusetzen und Servos, die noch nicht erkannt worden sind, neu zu erkennen. (Beispiel: Wenn der Buchstabe A in dem Wort **AXISSTATUS** rot dargestellt wird).
MECHATROLINK(baugruppe,3,VR)
Gibt die Anzahl der erkannten MECHATROLINK-II-Geräte nach einem **MECHATROLINK(baugruppe,0)** -Befehl zurück. Er wird vom **STARTUP**-Programm verwendet, um zu überprüfen, ob die Anzahl der erkannten MECHATROLINK-II-Module und mit der erwarteten Anzahl übereinstimmt.
MECHATROLINK(baugruppe,4,station,VR)
Gibt die Adresse des MECHATROLINK-II-Gerätes an dieser „Stations“-Nummer zurück. Die Stationsnummern sind bei allen angeschlossenen Geräten eine Folge von 0..x. –1 wird zurückgegeben, wenn der betreffenden Station kein Gerät zugeordnet ist. Er wird vom **STARTUP**-Programm verwendet, um zu überprüfen, ob die Anzahl der erkannten MECHATROLINK-II-Module und mit der erwarteten Anzahl übereinstimmt.
MECHATROLINK(baugruppe,5,station,VR)
Liest und löscht den Zähler für fehlende Meldungen. Ein Nicht-Achsen-MECHATROLINK-II-Gerät meldet ein Netzwerkproblem nicht automatisch, daher kann dieser Befehl dazu verwendet werden, den Frequenzumrichter und die E/A-Module abzufragen, um zu überprüfen, ob das Netzwerk funktioniert. Dieser Befehl wird in dem vorgeschlagenen SHELL-Programm verwendet.
Hinweis:

- Sie können den Befehl **MECHATROLINK(baugruppe,5,station,VR)** dazu verwenden, den Zustand eines Geräts während einer Programmausführung zu überwachen.
  Wenn der Inhalt der VR-Adresse größer als 0 ist, tritt ein Kommunikationsfehler mit dem Gerät auf, und es kann eine Fehlfunktion am Gerät entstehen. Sie können diesen Befehl dazu verwenden, das Programm zu stoppen, wenn das Gerät einen Fehler aufweist.

Beschreibung Hinweis: Dieser Befehl hat zwei Formen, abhängig von der benötigten Funktion: Master- und Stationsfunktionen.
Alle **MECHATROLINK** -Funktionen geben **TRUE** (–1) zurück, wenn der Befehl erfolgreich war oder **FALSE** (0), wenn der Befehl fehlgeschlagen ist.
Die Funktionen sind in zwei Typen unterteilt, **MASTER**-Funktionen, die sich auf eine Baugruppe auswirken und **STATION**-Funktionen, die sich auf eine bestimmte **Stations_adresse** einer gegebenen Baugruppe auswirken.
Alle Funktionen, die einen Wert abrufen, speichern ihn in der **VR**-Variablen, die im letzten Parameter angegeben ist. Wenn dieser Parameter den Wert –1 hat, wird der Wert an der Befehlszeilenschnittstelle ausgegeben.
Hinweise:

- Wenn ein MECHATROLINK-II-Befehl fehlschlägt, geht die MECHATROLINK-II-Station in einen Warn-/Alarmzustand. Alle nachfolgenden Befehle geben dann diesen Warn-/Alarmzustand zurück, auch wenn der betreffende Befehl korrekt ausgeführt wird.
- Der Warn-/Alarmzustand kann nur durch den Befehl **ALM_CLR** gelöscht werden.
- Es gibt keinen **ALM_CLR** -Unterbefehl, daher muss der Inbetriebnahme-Modus aufgerufen werden, um den Befehl **ALM_CLR** zu senden.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.173 MERGE

Typ Achsenparameter

Syntax **MERGE**

Beschreibung Der Parameter **MERGE** ist ein Software-Schalter, der zum Aktivieren oder Deaktivieren der Verknüpfung aufeinander folgender Bewegungen verwendet werden kann. Wenn **MERGE** aktiviert und die nächste Bewegung bereits im Bewegungs-Befehlsspeicher (**NTYPE**) ist, verzögert die Achse nicht bis auf Nulldrehzahl, sondern führt die folgende Bewegung aus, was einen nahtlosen Übergang ermöglicht. Die Standardeinstellung von **MERGE** ist AUS.
Der Programmierer muss sicherstellen, dass die Übergänge sinnvoll sind. Beispielsweise hätte die Verknüpfung einer Vorwärtsbewegung mit einer Rückwärtsbewegung den Versuch einer abrupten Richtungsänderung zur Folge.
MERGE funktioniert nur, wenn alle folgenden Bedingungen erfüllt sind:

1. Nur die Drehzahl Profil-Bewegungen **MOVE**, **MOVEABS**, **MOVECIRC**, **MHELICAL**, **REVERSE**, **FORWARD** und **MOVEMODIFY** können miteinander verbunden werden. Sie können nicht mit den synchronisierten Bewegungen **CONNECT**, **MOVELINK** und **CAMBOX** verbunden werden.
2. Es befindet sich eine Bewegung im Befehlsspeicher „Nächste Bewegung“ (**NTYPE**).
3. Die Achsengruppe ändert sich nicht bei Bewegungen mit mehreren Achsen.

Wenn Bewegungen mit mehreren Achsen miteinander verbunden werden, braucht **MERGE** nur für den Achsenparameter der Basisachse eingestellt zu werden.
Hinweis: Wenn die Bewegungen kurz sind, muss eine hohe Verzögerungsrate eingestellt werden, um zu vermeiden, dass der TJ1-MC__ in Erwartung des Endes der gespeicherten Bewegung verzögert.

Argumente –

Beispiel **MERGE = OFF ' Verzögert nach jeder einzelnen Bewegung**
MERGE = ON ' Bewegungen werden miteinander verbunden, falls möglich

Siehe auch **AXIS**.

3.2.174 MHELICAL

Typ	Achsenbefehl
Syntax	**MHELICAL(ende1, ende2, mittelpunkt1, mittelpunkt2, richtung, distanz3)** **MH(ende1, ende2, mittelpunkt1, mittelpunkt2, richtung, distanz3)**
Beschreibung	Führt eine spiralförmige Bewegung aus, d. h. bewegt zwei orthogonale Achsen so, dass ein kreisförmiger Bogen an der Werkzeugspitze mit einer gleichzeitigen linearen Bewegung über eine dritte Achsen beschrieben wird. Die ersten 5 Parameter sind ähnlich wie bei einem **MOVECIRC()**-Befehl. Der sechste Parameter definiert die gleichzeitige lineare Bewegung. Ende 1 und Mittelpunkt 1 sind an der aktuellen **BASE**-Achse. Ende 2 und Mittelpunkt 2 sind an der darauf folgenden Achse. Die ersten 4 Distanzen und der sechste Parameter werden nach dem aktuellen Einheiten-Umwandlungsfaktor für jede Achse skaliert.
Argumente	• **ende1** Endposition an der **BASE**-Achse. • **ende2** Endposition an der nächsten Achse im **BASE**-Array. • **mittelpunkt1** Position an der **BASE**-Achse, um die gedreht werden soll. • **mittelpunkt2** Position an der nächsten Achse im **BASE**-Array, um die gedreht werden soll. • **richtung** **richtung** ist ein Software-Schalter, mit dem ermittelt wird, ob der Bogen im Uhrzeigersinn oder gegen den Uhrzeigersinn interpoliert werden soll. Der Parameter wird auf 0 oder 1 gesetzt. Siehe **MOVECIRC**. • **distanz3** Die Distanz, die auf der dritten Achse im **BASE**-Array zurückgelegt werden soll, angegeben in Benutzereinheiten.
Beispiel	Kein Beispiel.
Siehe auch	**MOVECIRC**.

3.2.175 MOD

Typ	Mathematische Funktion
Syntax	**ausdruck1 MOD ausdruck2**
Beschreibung	Die Funktion **MOD** gibt den **ausdruck2**-Modulus von **ausdruck1** zurück. Diese Funktion benutzt den Integer-Teil einer Nicht-Integer-Eingabe.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT 122 MOD 13** **5.0000**
Siehe auch	–

3.2.176 MOTION_ERROR

Typ	Systemparameter (Nur-Lesen)
Syntax	**MOTION_ERROR**
Beschreibung	Der Parameter **MOTION_ERROR** enthält ein Bit-Muster, dass die Achsen anzeigt, die einen Bewegungsfehler haben. Wenn z. B. die Achsen 2 und 6 einen Bewegungsfehler haben, wäre der Wert für MOTION_ERROR 68 (4 + 64). Ein Bewegungsfehler tritt auf, wenn der **AXISSTATUS**-Zustand für eine der Achsen mit der Einstellung **ERRORMASK** übereinstimmt. In diesem Fall wird die Achsenfreigabe (**WDOG**) ausgeschaltet, und **MOTION_ERROR** enthält ein Bit-Muster, das alle Achsen anzeigt, die den Bewegungsfehler haben; der Parameter **ERROR_AXIS** enthält die Nummer der ersten Achse, die den Fehler hat. Ein Bewegungsfehler kann mit dem Befehl **DATUM(0)** gelöscht werden.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **AXISSTATUS**, **DATUM**, **ERROR_AXIS**, **ERRORMASK**, **WDOG**.

3.2.177 MOVE

Typ Achsenbefehl

Syntax **MOVE(distanz_1 [, distanz_2 [, distanz_3 [, distanz_4 [, ...]]]])**
MO(distanz_1 [, distanz_2 [, distanz_3 [, distanz_4 [, ...]]]])

Beschreibung Der Befehl **MOVE** bewegt eine oder mehrere Achsen mit der gewünschten Geschwindigkeit, Beschleunigung und Verzögerung zu einer Position, die als Inkrement von der aktuellen Position spezifiziert ist. Bei Bewegungen mit mehreren Achsen wird die Bewegung interpoliert; die Drehzahl, Beschleunigung und Verzögerung werden von der Basisachse übernommen. Die spezifizierten Distanzen werden mit dem im Achsenparameter **UNITS** angegebenen Umwandlungsfaktor skaliert. Wenn z. B. eine Achse 4000 Drehgeber-Flanken/mm hat, wird die Anzahl der Einheiten für diese Achse auf 4000 gesetzt, und **MOVE**(12,5) wäre 12,5 mm.
MOVE wirkt sich auf die Basisachsengruppe aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Das Argument **distanz_1** wird auf die Basisachse angewendet, **distanz_2** auf die nächste Achse usw. Durch Wechseln der Achsen zwischen einzelnen **MOVE**-Befehlen können uninterpolierte, unsynchronisierte Bewegungen mit mehreren Achsen erreicht werden. Inkrementale Bewegungen können durch Aktivieren des Achsenparameters **MERGE** zu stetigen, positionssynchronisierten Bewegungen verbunden werden.
Bei einer Bewegung mit zwei Achsen werden die einzelnen Drehzahlen mit den unten stehenden Gleichungen berechnet. Gegeben ist der Befehl **MOVE(**x_1,x_2**)** und die Profildrehzahl v_p, wie berechnet aus den Parametern **SPEED**, **ACCEL** und **DECEL** von der Basisachse und der Gesamtdistanz von mehreren Achsen L = **SQR(**$x_1^2 + x_2^2$**)**.
Die individuelle Drehzahl v_i für Achse i wird zu jedem beliebigen Zeitpunkt der Bewegung berechnet als: $v_i = (x_i * v_p)/L$.

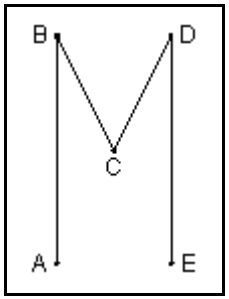
Argumente Der Befehl kann bis zu 16 Argumente annehmen.

- **distanz_i**
  Die Distanz in Benutzereinheiten, die für jede Achse *i* zurückgelegt werden soll, angefangen mit der Basisachse.

Beispiel Ein System arbeitet mit einem Einheiten-Umwandlungsfaktor von 1 und hat einen 1000-Line-Drehgeber. Um eine Bewegung von 10 Motordrehungen auszuführen, ist folgender Befehl erforderlich. (Ein 1000-Line-Drehgeber erzeugt 4000 Flanken/Umdrehung).
MOVE(40000)

Beispiel
```
MOVE(10) AXIS(0)
MOVE(10) AXIS(1)
MOVE(10) AXIS(2)
```
In diesem Beispiel werden die Achsen 0, 1 und 2 unabhängig voneinander bewegt (ohne Interpolation). Jede Achse bewegt sich mit ihrer programmierten Drehzahl und anderen Achsenparametern.

Beispiel Ein X-Y-Plotter kann innerhalb seines Arbeitsraumes Text an jede beliebige Position schreiben. Einzelne Zeichen sind als Folge von Bewegungen definiert, die relativ zum Startpunkt erfolgen, sodass die gleichen Befehle ungeachtet der Plot-Position benutzt werden können. Das Befehls-Unterprogramm für den Buchstaben M könnte wie folgt aussehen:
```
MOVE(0,12) ' A -> B
MOVE(3,–6) ' B -> C
MOVE(3,6) ' C -> D
MOVE(0,–12) ' D -> E
```



Siehe auch **AXIS**, **MOVEABS**, **UNITS**.

3.2.178 MOVEABS

Typ Achsenbefehl

Syntax **MOVEABS(distanz_1 [, distanz_2 [, distanz_3 [, distanz_4 [, ...]]]])**
MA(distanz_1 [, distanz_2 [, distanz_3 [, distanz_4 [, ...]]]])

Beschreibung Der Befehl **MOVEABS** bewegt eine oder mehrere Achsen mit der gewünschten Geschwindigkeit, Beschleunigung und Verzögerung zu einer Position, die als Absolutposition spezifiziert ist, d. h. in Bezug zum Ausgangspunkt. Bei Bewegungen mit mehreren Achsen wird die Bewegung interpoliert; die Drehzahl, Beschleunigung und Verzögerung werden von der Basisachse übernommen.
Die spezifizierten Distanzen werden mit dem im Achsenparameter **UNITS** angegebenen Umwandlungsfaktor skaliert. Wenn z. B. eine Achse 4000 Drehgeber-Flanken/mm hat, wird die Anzahl der Einheiten für diese Achse auf 4000 gesetzt, und **MOVEABS**(12,5) würde eine Bewegung zu einer Position 12,5 mm vom Ausgangspunkt ausführen. **MOVEABS** wirkt sich auf die standardmäßige Achsenbasis aus (eingestellt mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Das Argument **distanz_1** wird auf die Basisachse angewendet, **distanz_2** auf die nächste Achse usw. Durch Wechseln der Achsen zwischen einzelnen **MOVE**-Befehlen können uninterpolierte, unsynchronisierte Bewegungen mit mehreren Achsen erreicht werden. Absolute Bewegungen können durch Aktivieren des Achsenparameters **MERGE** zu stetigen, positionssynchronisierten Bewegungen verbunden werden.
Bei einer Bewegung mit zwei Achsen werden die einzelnen Drehzahlen mit den unten stehenden Gleichungen berechnet. Gegeben ist der Befehl **MOVE(**ax_1,ax_2**)**, die aktuelle Position (ay_1,ay_2) und die Profildrehzahl v_p, wie berechnet aus den Parametern **SPEED**, **ACCEL** und **DECEL** von der Basisachse und der Gesamtdistanz von mehreren Achsen L = **SQR(**$x_1^2 + x_2^2$**)**, wobei $x_1 = ax_i - ay_i$.
Die individuelle Drehzahl für eine Achse lässt sich zu jedem Zeitpunkt der Bewegung berechnen aus $v_i = (x_i \times v_p)/L$.

Argumente Der Befehl kann bis zu 16 Argumente annehmen.

- **distanz_i**
  Die Position in Benutzereinheiten, zu der jede Achse *i* bewegt werden soll, angefangen mit der Basisachse.

Beispiel **MOVEABS(20,350)**
Ein X-Y-Plotter hat ein Stiftkarussel, dessen Position relativ zum Plotter-Ausgangspunkt fixiert ist. Um den Stift zu wechseln, kann, ungeachtet der aktuellen Plot-Position, mit einer absoluten Bewegung zur Karussellposition gefahren werden.

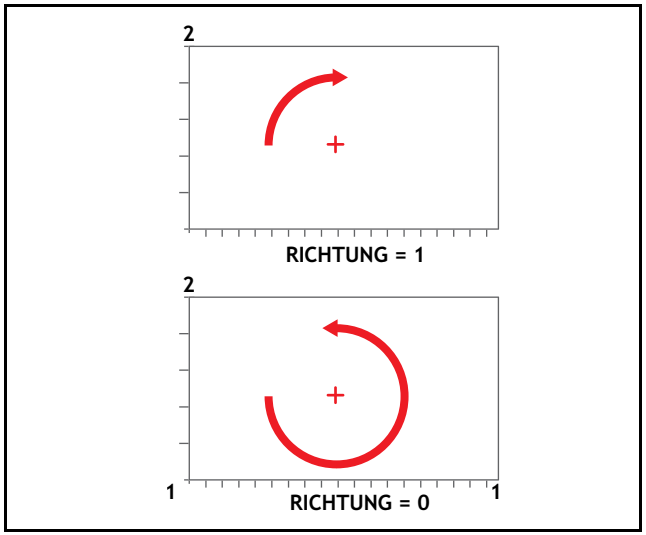
Beispiel Eine Palette besteht aus einem 6 mal 8-Gitter, in welches Gasflaschen mit einem Abstand von 85 mm voneinander von einer Packmaschine eingesteckt werden. Die Flaschen werden an einem festgelegten Punkt aufgenommen. Die erste Position in der Palette ist mit dem Befehl DEFPOS als Position (0,0) definiert. Der Programmteil zur Positionierung der Flaschen in der Palette sieht wie folgt aus:

```
xschleife:
  FOR x = 0 TO 5
yschleife:
    FOR y = 0 TO 7
    MOVEABS(–340,–516.5) ' Bewegung zum Aufnahmepunkt
    GOSUB pick ' Zum Aufnahme-Unterprogramm gehen
    PRINT "BEWEGUNG ZU POSITION: ";x * 6 + y + 1
    MOVEABS(x * 85, y * 85)
    GOSUB place ' Zum Absetz-Unterprogramm gehen
    NEXT y
  NEXT x
```

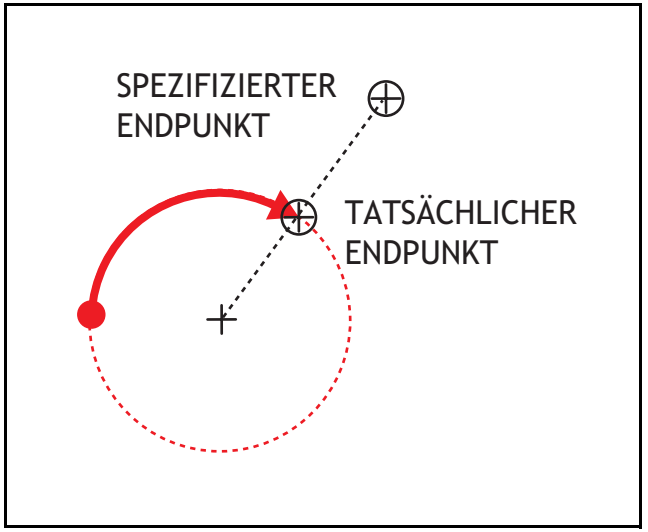
Siehe auch **AXIS**, **MOVE**, **MOVEABS**, **UNITS**.

3.2.179 MOVECIRC

Typ	Achsenbefehl
Syntax	**MOVECIRC(ende_1, ende_2, mittelpunkt_1, mittelpunkt_2, richtung)** **MOVECIRC(ende_1, ende_2, mittelpunkt_1, mittelpunkt_2, richtung)**
Beschreibung	Der Befehl **MOVECIRC** interpoliert 2 orthogonale Achsen in einem Kreisbogen. Der Pfad der Bewegung wird durch die fünf Argumente bestimmt, die inkrementell von der aktuellen Position aus erfolgen. Die Argumente **ende_1** und **mittelpunkt_1** wirken sich auf die Basisachse aus, **ende_2** und **mittelpunkt_2** auf die folgenden Achsen. Alle Argumente werden bei jeder Achse in Benutzereinheiten angegeben. Die Geschwindigkeit der Bewegung im Kreisbogen wird durch die Parameter **SPEED**, **ACCEL** und **DECEL** der Basisachse bestimmt. **MOVECIRC** wirkt sich auf die Basisachsengruppe aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Damit **MOVECIRC** korrekt ausgeführt wird, müssen die beiden Achsen, die sich im Kreisbogen bewegen, die gleiche Anzahl an Drehgeberimpulsen pro lineare Achsendistanz haben. Ist das nicht der Fall, ist es in vielen Fällen möglich, die Drehgeber-Skalierung durch Einstellen der Achsenparameter **ENCODER_RATIO** für die Achse anzupassen.
Argumente	• **ende_1** Die Endposition für die Basisachse. • **ende_2** Die Endposition für die nächste Achse. • **mittelpunkt_1** Die Position, um die die Basisachse gedreht werden soll. • **mittelpunkt_2** Die Position, um die die nächste Achse gedreht werden soll. • **richtung** Ein Software-Schalter, mit dem ermittelt wird, ob der Bogen im Uhrzeigersinn oder gegen den Uhrzeigersinn interpoliert werden soll. Wert: 0 oder 1. Wenn die beiden an der Bewegung beteiligten Achsen ein rechtshändiges System bilden, stellen Sie die Richtung auf 0 ein, um eine positive Bewegung um die dritte (möglicherweise imaginäre) orthogonale Achse zu erreichen. Wenn die beiden an der Bewegung beteiligten Achsen ein linkshändiges System bilden, stellen Sie die Richtung auf 0 ein, um eine negative Bewegung um die dritte (möglicherweise imaginäre) orthogonale Achse zu erhalten. Siehe Tabelle unten.



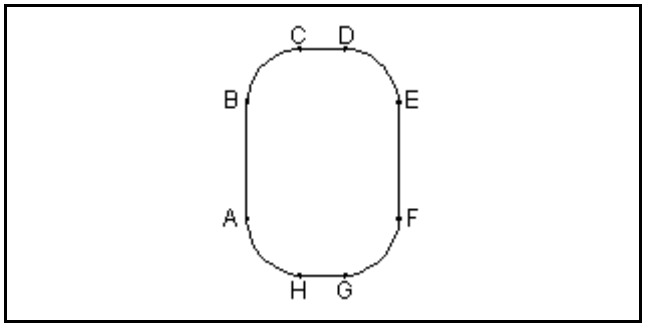
Hinweis: Mit dem Befehl **MOVECIRC** werden der Radius und der gesamte Drehwinkel ab dem Mittelpunkt bis zum Endpunkt berechnet. Wenn der Endpunkt nicht auf dem berechneten Pfad liegt, endet die Bewegung einfach an dem berechneten Ende und nicht an dem spezifizierten Endpunkt.
Der Programmierer muss sicherstellen, dass die beiden Punkte mit den richtigen Punkten auf einem Kreis übereinstimmen.



Beispiel

Die Befehlssequenz, um den Buchstaben 0 zu plotten, könnte wie folgt aussehen:

```
MOVE(0,6) ' Bewegung A -> B
MOVECIRC(3,3,3,0,1) ' Bewegung B -> C
MOVE(2,0) ' Bewegung C -> D
MOVECIRC(3,–3,0,–3,1) ' Bewegung D -> E
MOVE(0,–6) ' Bewegung E -> F
MOVECIRC(–3,–3,–3,0,1) ' Bewegung F -> G
MOVE(–2,0) ' Bewegung G -> H
MOVECIRC(–3,3,0,3,1) ' Bewegung H -> A
```



Siehe auch **AXIS**, **ENCODER_RATIO**, **UNITS**

Richtung	Rechte Achse	Linke Achse
1	Negativ	Positiv
0	Positiv	Negativ

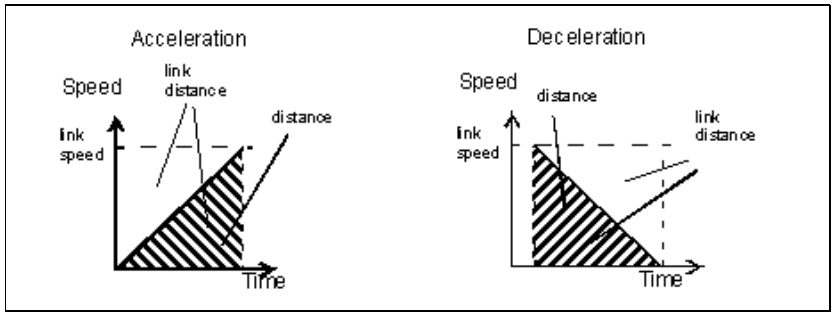
3.2.180 MOVELINK

Typ Achsenbefehl

Syntax **MOVELINK(strecke, link_strecke, link_beschleunigung, link _verzoegerung, link _achse [, link_option [, link_position]])**
ML(strecke, link_strecke, link_beschleunigung, link _verzoegerung, link _achse [, link_option [, link_position]])

Beschreibung Der Befehl **MOVELINK** erzeugt eine lineare Bewegung der Basisachse, die über ein Software-Getriebe mit der gemessenen Position einer synchronisierten Achse verbunden ist. Die synchronisierte Achse kann sich in beide Richtungen bewegen, um die Ausgangsbewegung abzufahren. Die Parameter zeigen die Strecke, die sich die Basisachse für eine bestimmte Strecke der synchronisierten Achse (**link_ strecke**) bewegt. Die Strecke der synchronisierten Achse ist in drei Phasen unterteilt, die sich auf die Bewegung der Basisachse auswirken. Diese Teile sind die Beschleunigung, die konstante Geschwindigkeit und die Verzögerung. Die synchronisierten Beschleunigungs- und Verzögerungsdistanzen werden durch die Parameter **link_beschleunigung** und **link_verzoegerung** spezifiziert. Die synchronisierte Strecke mit konstanter Drehzahl wird von der gesamten synchronisierten Strecke und diesen beiden Parametern abgeleitet.
Die drei Phasen können in einzelne **MOVELINK** -Befehle unterteilt oder zu einer addiert werden.
Beachten Sie bei der Einstellung des **MOVELINK**-Befehls die folgenden beiden Regeln:
Regel 1: In der Beschleunigungs- und Verzögerungsphase mit übereinstimmender Drehzahl muss die **link_strecke** das Doppelte der Strecke betragen. Siehe Abbildung.
Regel 2: In der Phase mit übereinstimmender Drehzahl müssen die beiden Achsen dieselbe Strecke zurücklegen deshalb muss die zurückgelegte Strecke gleich der **link_strecke** sein.
MOVELINK wirkt sich auf die Basisachsengruppe aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren. Die als **link_achse** eingestellte Achse treibt die Basisachse an.
Hinweis: Wenn die Summe aus **link_beschleunigung** und **link _verzoegerung** größer ist als die **link_strecke**, werden sie beide proportional verringert, damit die Summe gleich der **link_strecke** ist.

Argumente



- **strecke**
  Die inkrementelle Strecke in Benutzereinheiten, die sich die Basisachse bewegen soll, als Ergebnis der an der synchronisierten Achse gemessenen Bewegung über die **link_strecke**.
- **link_strecke**
  Die positive inkrementelle Strecke in Benutzereinheiten, die an der synchronisierten Achse gemessen werden muss, damit sich die Basisachse um die gewünschte Strecke bewegt.
- **link_beschleunigung**
  Der positive Weg in Benutzereinheiten an der Masterachse, innerhalb der die Slaveachse beschleunigen soll.
- **link_verzoegerung**
  Der positive Weg in Benutzereinheiten an der synchronisierten Achse, innerhalb der die Basisachse verzögern soll.
  Hinweis: Wenn die Summe aus Parameter 3 und Parameter 4 größer ist als Parameter 2, werden beide proportional verringert, bis sie gleich Parameter 2 sind.
- **link_achse**
  Die Achse, mit der synchronisiert werden soll.
- **link_option**
  Siehe Tabelle unten.
- **link_position**
  Die Absolutposition, bei der **MOVELINK** startet, wenn **link_option** auf 2 gesetzt ist.

Hinweis: Der Befehl verwendet BASE() und AXIS() sowie Einheiten-Umwandlungsfaktoren in ähnlicher Weise wie andere MOVE-Befehle.
Hinweis: Die „synchronisierte“ Achse kann sich in beide Richtungen bewegen, um die Ausgangsbewegung abzufahren. Die synchronisierten Strecken sind immer positiv.

Beispiel

Ein Papierschneider schneidet eine Rolle Papier alle 160 m, wobei er sich mit der Geschwindigkeit des Papiers bewegt. Das Schneidwerkzeug kann bis zu 1,2 m weit fahren, wovon in diesem Beispiel 1 m genutzt wird.
Die Papierposition wird von einem Drehgeber gemessen, der Einheiten-Umwandlungsfaktor ist so eingestellt, dass bei beiden Achsen die Einheit Meter ist. Achse 1 ist die synchronisierte Achse.

```
MOVELINK(0,150,0,0,1) ' Startposition abwarten
MOVELINK(0.4,0.8,0.8,0,1) ' beschleunigen
MOVELINK(0.6,1.0,0,0.8,1) ' konstante Drehzahl, dann verzögern
WAIT UNTIL NTYPE = 0 ' warten, bis letzte Bewegung gestartet ist
OP(8,ON) ' Schneidwerkzeug aktivieren
MOVELINK(–1,8.2,0.5,0.5,1) ' zurückfahren
```

In diesem Programm wartet der TJ1-MC__, bis 150 m von der Rolle abgezogen sind. Bei dieser Position beginnt die Beschleunigung des Schneidwerkzeugs auf die Geschwindigkeit des Papiers. Dann läuft es mit der gleichen Geschwindigkeit wie das Papier, verzögert bis zum Stillstand und fährt mit einem Hub von 1 m zum Ausgangspunkt zurück. Diese Bewegung wird durch zwei getrennte **MOVELINK**-Befehle spezifiziert. Anschließend wartet das Programm darauf, dass der Befehlsspeicher für die nächste Bewegung leer wird (**NTYPE = 0**). Dies zeigt an, dass die Beschleunigungsphase abgeschlossen ist. Die Strecken auf der synchronisierten Achse (**link_strecke**) in den **MOVELINK**-Befehlen sind 150, 0,8, 1,0, und 8,2, was zusammen 160 m ergibt. Um sicherzustellen, dass die Geschwindigkeiten und Positionen des Schneidwerkzeugs und des Papiers während der Schneidaufgabe übereinstimmen, müssen die Argumente des Befehls **MOVELINK** korrekt sein.
Hinweis: Berücksichtigen Sie die Phasen für Beschleunigung, konstante Drehzahl und Verzögerung separat. Wie bereits erwähnt, muss in den Beschleunigungs- und Verzögerungsphasen die **link_strecke** das Doppelte der Distanz betragen. Beide Phasen können spezifiziert werden als:

```
MOVELINK(0.4,0.8,0.8,0,1) ' Beschleunigungsphase
MOVELINK(0.4,0.8,0,0.8,1) ' Verzögerungsphase
```

In einer Gleichlaufstrecke mit übereinstimmenden Drehzahlen legen die zwei Achsen den gleichen Weg zurück, deshalb muss die Strecke der Bewegung gleich der synchronisierten Strecke sein. Die Gleichlaufstrecke kann wie folgt spezifiziert werden:

```
MOVELINK(0.2,0.2,0,0,1) ' Gleichlaufstrecke
```

Der Befehl **MOVELINK** erlaubt die Addition der drei Abschnitte, indem für jede Phase die Werte für **strecke**, **link_strecke**, **link_beschleunigung** und **link_verzoegerung** zusammengezählt werden, wodurch sich folgender Befehl ergibt.

```
MOVELINK(1,1.8,0.8,0.8,1)
```

In dem obigen Programm wird die Beschleunigungsphase getrennt programmiert. Damit ist am Ende der Beschleunigungsphase eine Aktion möglich.

```
MOVELINK(0.4,0.8,0.8,0,1)
MOVELINK(0.6,1.0,0,0.8,1)
```

Siehe auch **AXIS**, **UNITS**, **REP_OPTION**.

Synchronisierungsoption	Beschreibung
1	Die Synchronisierung startet, wenn an der synchronisierten Achse eine Druckmarke erkannt wurde.
2	Die Synchronisierung startet an einer absoluten Position auf der synchronisierten Achse (siehe **link_position**).
4	**MOVELINK** wird automatisch bidirektional wiederholt. Diese Option wird aufgehoben, indem Bit 1 des Parameters **REP_OPTION** gesetzt wird (d. h. **REP_OPTION = REP_OPTION OR 2**).
5	Kombination der Optionen 1 und 4.
6	Kombination der Optionen 2 und 4.

3.2.181 MOVEMODIFY

Typ — Achsenbefehl

Syntax

```
MOVEMODIFY(position)
MM(position)
```

Beschreibung — Der Befehl **MOVEMODIFY** ändert die absolute Endposition der aktuellen einachsigen linearen Bewegung (**MOVE** oder **MOVEABS**). Wenn es keine aktuelle Bewegung gibt oder die aktuelle Bewegung keine lineare Bewegung ist, wird **MOVEMODIFY** als **MOVEABS**-Befehl behandelt. Der Achsenparameter **ENDMOVE** enthält die Position am Ende der aktuellen Bewegung in Benutzereinheiten.
MOVEMODIFY wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.

Argumente

- **Position**
  Die Absolutposition, die als das neue Ende der Bewegung eingestellt werden soll.

Beispiel — Kein Beispiel.

Siehe auch — **AXIS**, **MOVE**, **MOVEABS**, **UNITS**.

3.2.182 MPOS

Typ	Achsenparameter (Nur-Lesen)
Syntax	**MPOS**
Beschreibung	Der Parameter **MPOS** ist die gemessene Positionen der Achse in Benutzereinheiten, wie sie vom Drehgeber abgeleitet wird. Dieser Parameter kann über den Befehl **DEFPOS** eingestellt werden. Der Achsenparameter **OFFPOS** kann ebenfalls dazu benutzt werden, den Ausgangspunkt zu verschieben. **MPOS** wird beim Hochfahren auf 0 zurückgesetzt. Der Bereich der gemessenen Position wird mit den Achsenparametern **REP_DIST** und **REP_OPTION** kontrolliert.
Argumente	–
Beispiel	**WAIT UNTIL MPOS >= 1250** **SPEED = 2,5**
Siehe auch	**UNITS**, **AXIS**, **DEFPOS**, **ENCODER**, **FE**, **OFFPOS**, **REP_DIST**, **REP_OPTION**, **UNITS**.

3.2.183 MSPEED

Typ	Achsenparameter (Nur-Lesen)
Syntax	**MSPEED**
Beschreibung	Der Parameter **MSPEED** enthält die gemessene Drehzahl in Benutzereinheiten/s. Er wird berechnet, indem die Änderungen an der gemessenen Position im letzten Servozyklus in Benutzereinheiten durch die Servo Zykluszeit (in Sekunden) dividiert wird. Die Servo Zykluszeit wird mit dem Parameter **SERVO_PERIOD** eingestellt. **MSPEED** stellt den Momentanwert der Drehzahl mit erheblicher Schwankungsbreite dar, die besonders bei niedrigen Drehzahlen auftreten kann. Es kann angebracht sein, den Durchschnitt aus mehreren Messwerten zu errechnen, wenn ein stabiler Wert bei niedrigen Drehzahlen erforderlich ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **SERVO_PERIOD**, **VP_SPEED**, **UNITS**.

3.2.184 MTYPE

Typ	Achsenparameter (Nur-Lesen)
Syntax	**MTYPE**.
Beschreibung	Der Parameter **MTYPE** enthält den Bewegungsbefehl, der gerade ausgeführt wird. Die möglichen Befehle sind in der nachstehenden Tabelle aufgeführt. **MTYPE** kann verwendet werden, um zu ermitteln, ob eine Bewegung beendet ist oder ob ein Übergang von einer Bewegungsart zu einer anderen stattgefunden hat. Wenn MTYPE einen Befehl enthält, heißt das nicht unbedingt, dass sich die Achse tatsächlich bewegt. Sie kann im Verlauf einer Bewegung auf Nulldrehzahl sein oder mit einer anderen Achse interpoliert sein, ohne sich selbst zu bewegen.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **NTYPE**.

Bewegungsnummer	**Bewegungsart**
0	**IDLE**(keine Bewegung)
1	**MOVE**
2	**MOVEABS**
3	**MHELICAL**
4	**MOVECIRC**
5	**MOVEMODIFY**
10	**FORWARD**
11	**REVERSE**
12	**DATUM**
13	**CAM**
14	Informationen zu JOG_FORWARD finden Sie unter **FWD_JOG**
15	Informationen zu JOG_REVERSE finden Sie unter **REV_JOG**
20	**CAMBOX**

Bewegungsnummer	Bewegungsart
21	**CONNECT**
22	**MOVELINK**

3.2.185 NAIO

Typ	Systemparameter (Nur-Lesen)
Syntax	**NAIO**
Beschreibung	Dieser Parameter gibt die Anzahl der analogen Eingangskanäle zurück, die an den MECHATROLINK-II-Erweiterungsbus angeschlossen sind. Beispielsweise gibt ein TJ1-MC__ 8 zurück, wenn 2 x AN2900-Baugruppen angeschlossen sind, da jede vier analoge Eingangskanäle hat.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.186 NEG_OFFSET

Typ	Systemparameter
Syntax	**NEG_OFFSET = Wert**
Beschreibung	Für Piezo-Betrieb. Wendet einen negativen Offset auf das Ausgangs-**DAC**-Signal vom Servo-Regelkreis an. Der Offset wird nach der Funktion **DAC_SCALE** angewendet. Ein Offset von 327 entspricht einem Offset von 0,1 Volt. Es wird vorgeschlagen, als Offset 65 % bis 70 % des erforderlichen Wertes anzuwenden, um eine stufige Bewegung bei offenem Regelkreis zu erreichen.
Argumente	• **Wert** Ein BASIC-Ausdruck.
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.187 NEW

Typ	Programmbefehl
Syntax	**NEW ["Programm_name"]**
Beschreibung	Der Befehl **NEW** löscht alle Programmzeilen des Programms im Controller. **NEW** ohne Programmname kann dazu verwendet werden, das aktuell ausgewählte Programm zu löschen (mit **SELECT**). Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. **NEW ALL** löscht alle Programme. Der Befehl kann auch dazu verwendet werden, die Tabelle zu löschen. **NEW "TABLE"** Die Bezeichnung **"TABLE"** muss in Anführungszeichen gesetzt sein. Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann der Benutzer den Befehl aus dem Menü Programm auswählen.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**COPY**, **DEL**, **RENAME**, **SELECT**, **TABLE**

3.2.188 NEXT

Siehe **FOR..TO..STEP..NEXT**.

3.2.189 NIO

Typ	Systemparameter
Syntax	**NIO**
Beschreibung	Gibt die Anzahl der im System installierten oder an den MECHATROLINK-II-Erweiterungsbus angeschlossenen Eingänge/Ausgänge zurück. Ein TJ-MC__ ohne MECHATROLINK-II-E/A gibt **NIO = 32** zurück. Die integrierten Eingänge sind Kanäle von 0 bis 15. Die integrierten Ausgänge sind Kanäle von 8 bis 15. Die Kanäle 16 bis 27 können als „virtuelle“ E/A verwendet werden, die zusammengeschlossen sind. Die Eingangskanäle 28 bis 31 sind dafür reserviert, dass die einzelnen Achsen die MECHATROLINK-II-Treiber-Eingangskanäle für Achsensteuerungsfunktionen verwenden können.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.190 NOT

Typ	Mathematische Operation
Syntax	**NOT Ausdruck**
Beschreibung	Der Operator **NOT** führt eine logische **NOT**-Verknüpfung aller Bits des Integer-Teils des Ausdrucks durch. Die logische **NOT**-Funktion ist in der Tabelle unten definiert.
Argumente	• **Ausdruck.** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT 7 AND NOT 1** **6.0000**
Siehe auch	–

Bit	Ergebnis
0	1
1	0

3.2.191 NTYPE

Typ	Achsenparameter (Nur-Lesen)
Syntax	**NTYPE**
Beschreibung	Der Parameter **NTYPE** enthält den nächsten Bewegungsbefehl. Wenn die aktuelle Bewegung beendet ist, wird die im **NTYPE**-Befehlsspeicher gespeicherte Bewegung ausgeführt. Die Werte sind die gleichen wie für den Achsenparameter **MTYPE**. **NTYPE** wird mit dem Befehl **CANCEL(1)** gelöscht.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **MTYPE**.

3.2.192 OFF

Typ	Konstante (Nur-Lesen)
Syntax	**OFF**
Beschreibung	Die Konstante **OFF** gibt den numerischen Wert 0 zurück.
Argumente	–
Beispiel	**OP (hebel,OFF)** **Die obige Zeile setzt den Ausgang mit der Bezeichnung 'hebel' auf AUS.**
Siehe auch	–

3.2.193 OFFPOS

Typ	Achsenparameter
Syntax	**OFFPOS**
Beschreibung	Der Parameter **OFFPOS** enthält einen Offset, der auf die Sollposition (**DPOS**) angewendet wird, ohne die Bewegung in anderer Weise zu beeinflussen. Die gemessene Position wird entsprechend geändert, um den Schleppfehler aufrechtzuerhalten. **OFFPOS** stellt wirkungsvoll die Nullposition der Achse ein. Der in **OFFPOS** eingestellte Wert wird vom System auf 0 zurückgesetzt, wenn der Offset geladen wird. Hinweis: Der Offset wird erst im nächsten Servozyklus angewendet. Vor dem nächsten Servozyklus können andere Befehle ausgeführt werden. Stellen Sie sicher, dass diese Befehle nicht davon ausgehen, dass die Positionsverschiebung erfolgt ist. Hierzu kann die Anweisung **WAIT UNTIL** verwendet werden (siehe Beispiel).
Argumente	–
Beispiel	Die folgenden Zeilen definieren die aktuelle Sollposition als 0. **OFFPOS = –DPOS** **WAIT UNTIL OFFPOS = 0 ' Warten bis zur Anwendung** Dieses Beispiel entspricht DEFPOS(0).
Siehe auch	**AXIS**, **DEFPOS**, **DPOS**, **MPOS**, **UNITS**.

3.2.194 ON

Typ	Konstante (Nur-Lesen)
Syntax	**ON**
Beschreibung	Die Konstante **ON** gibt den numerischen Wert 1 zurück.
Argumente	–
Beispiel	**OP (Hebel,ON)** **Die obige Zeile setzt den Ausgang mit der Bezeichnung 'hebel' auf EIN.**
Siehe auch	–

3.2.195 ON.. GOSUB

Typ	Programmsteuerungsbefehl
Syntax	**ON Ausdruck GOSUB Label { , Label }**
Beschreibung	Die Strukturen **ON..GOSUB** und **ON..GOTO** ermöglichen einen bedingten Sprung. Der Integer-Ausdruck wird verwendet, um einen Label aus der Liste auszuwählen. Hat der Ausdruck den Wert 1, wird das erste Label verwendet, bei dem Wert 2 das zweite Label usw. Wenn der Label ausgewählt ist, wird mit dem Unterprogramm **GOSUB** ein Sprung zu diesem Label ausgeführt. Hinweis: Wenn der Ausdruck nicht gültig ist, wird kein Sprung ausgeführt.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck. • **Label** Jedes gültige Label im Programm.
Beispiel	**REPEAT** **GET#5,Zeich** **UNTIL 1 <= Zeich und Zeich <= 3** **ON Zeich GOSUB beweger, stopper, wechseln**
Siehe auch	**GOSUB..RETURN**, **GOTO**.

3.2.196 ON.. GOTO

Typ	Programmsteuerungsbefehl
Syntax	**ON Ausdruck GOTO Label[,Label[,...]]**
Beschreibung	Der Ausdruck wird ausgewertet und dann der Integer-Teil verwendet, um ein Label aus der Liste auszuwählen. Hat der Ausdruck den Wert 1, wird das erste Label verwendet, bei dem Wert 2 das zweite Label usw. Wenn der Wert des Ausdrucks kleiner als 1 oder größer als die Anzahl der Labels ist, wird ein Fehler ausgegeben. Wenn das Label ausgewählt ist, wird mit dem Unterprogramm **GOTO** ein Sprung zu diesem Label ausgeführt.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck. • **Label** Jedes gültige Label im Programm.
Beispiel	**REPEAT** **GET #1,Zeich** **UNTIL 1 <= Zeich und Zeich <= 3** **ON Zeich GOTO beweger, stopper, wechseln**
Siehe auch	–

3.2.197 OP

Typ	E/A-Befehl
Syntax	**OP(ausgangs_nummer, wert)** **OP(binaer_muster)** **OP**

Beschreibung Der Befehl **OP** stellt einen oder mehrere Ausgänge ein oder gibt den Zustand der ersten 24 Ausgänge zurück. **OP** hat drei verschiedene Formen, abhängig von der Anzahl der Argumente.

- Der Befehl **OP(ausgangs_nummer,wert)** stellt einen einzelnen Ausgangskanal ein. Der Bereich für die **ausgangs_nummer** liegt zwischen 8 und 256; **wert** ist der auszugebende Wert, entweder 0 oder 1.
- Der Befehl **OP(binaer_muster)** stellt das binäre Muster für die 24 Ausgänge gemäß dem Wert ein, der durch **binaer_muster** vorgegeben ist.
- Die Funktion **OP (ohne Argumente)** gibt den Zustand der ersten 24 Ausgänge zurück. Dadurch ist es möglich, mehrere Ausgänge einzustellen, ohne andere zu beeinträchtigen, die nicht geändert werden sollen.

Hinweis: Die ersten 8 Ausgänge (0 bis 7) existieren physikalisch nicht im TJ1-MC__. Sie können nicht verändert werden, und sie geben immer 0 zurück.

Argumente

- **ausgangs_nummer**
  Die Nummer des Ausgangs, der eingestellt werden soll.
- **wert**
  Der Wert des Ausgangs, EIN oder AUS. Alle Werte, die nicht 0 sind, werden als EIN definiert.
- **binaer_muster**
  Das Integer-Äquivalent des Binärmusters, das ausgegeben werden soll.

Beispiel

```
OP(12,1)
OP(12,ON)
```

Diese beiden Zeilen sind äquivalent.

Beispiel

```
OP(18 * 256)
```

Diese Zeile stellt das Bitmuster 10010 auf die ersten 5 physikalischen Ausgänge ein, die Ausgänge 13 bis 17 würden gelöscht. Das Bitmuster wird durch Multiplikation mit 256 um 8 Bit verschoben, um die ersten verfügbaren Ausgänge einzustellen, da die Ausgänge 0 bis 7 nicht existieren.

Beispiel

```
VR(0) = OP
VR(0) = VR(0) AND 65280
OP(VR(0))
```

Diese Routine setzt die Ausgänge 8 bis 15 auf **EIN**, alle anderen auf AUS. Die obige Programmierung kann auch wie folgt geschrieben werden:
OP(OP AND 65280)

Beispiel

```
val = 8 ' Der einzustellende Wert
Maske = OP AND NOT(15 * 256) ' Liest aktuellen Status und Maske
OP(maske OR wert * 256) ' Stellt wert auf OP(8) bis OP(11)
```

Diese Routine stellt den Wert **Wer** für die Ausgänge 8 bis 11 ein, ohne die anderen Ausgänge zu beeinträchtigen, indem eine Maske verwendet wird.

Siehe auch **IN**.

3.2.198 OPEN_WIN

Typ Achsenparameter

Syntax **OPEN_WIN**
OW

Beschreibung Der Achsenparameter **OPEN_WIN** definiert den Anfang eines Fensters, in dem oder außerhalb von dem eine Druckmarke erwartet wird. Der Wert wird in Benutzereinheiten angegeben.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **CLOSE_WIN**, **REGIST**, **UNITS**.

3.2.199 OR

Typ Mathematische Operation

Syntax **ausdruck1 OR ausdruck2**

Beschreibung Der Operator **OR** führt eine logische **OR**-Verknüpfung zwischen entsprechenden Bits der Integer-Teile von zwei gültigen BASIC-Ausdrücken durch.
Die logische **OR**-Verknüpfung zwischen zwei Bits ist in der Tabelle unten definiert.

Argumente

- **ausdruck1**
  Jeder gültige BASIC-Ausdruck.
- **ausdruck2**
  Jeder gültige BASIC-Ausdruck.

Beispiel **Beispiel 1:**
Ergebnis = 10 OR (2.1 * 9)
Die Klammern werden zuerst berechnet, aber nur der Integer-Teil des Ergebnisses, 18, wird für die OR-Operation benutzt. Deshalb ist der Ausdruck gleich Folgendem:
Ergebnis = 10 OR 18

Deshalb enthält das Ergebnis den Wert 26.
Beispiel 2:

Beispiel **Ergebnis = 10 OR 18**
Das **OR** ist ein Bit-Operator, daher sieht der Binärcode so aus:
01010 OR 10010 = 11010

Beispiel **IF KEY OR VR(0) = 2 THEN GOTO label**

Siehe auch –

Bit 1	Bit 2	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	1

3.2.200 OUTDEVICE

Typ E/A-Parameter

Syntax **OUTDEVICE**

Beschreibung Der Parameter **OUTDEVICE** definiert das Standard-Ausgabegerät. Dieses Gerät wird für den Befehl **PRINT** ausgewählt, wenn die Option **#n** ausgelassen wird. Der Parameter **OUTDEVICE** ist Task-spezifisch.
Die unterstützten Werte sind in der nachstehenden Tabelle aufgeführt.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **PRINT**.

Wert	Beschreibung
0	Programmierschnittstelle 0 (Standard)
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.201 OUTLIMIT

Typ Achsenparameter

Syntax **OUTLIMIT**

Beschreibung Der Ausgabegrenzwert beschränkt die Sollwertausgabe einer Servoachse auf einen Wert, der unter dem Maximum liegt. Der benötigte Wert richtet sich nach der maximal möglichen Sollwertausgabe. Wenn der Spannungsausgang von einem 16-Bit DAC generiert wird, ergibt ein **OUTLIMIT**-Wert von 32767 den vollen +/–10 V-Bereich. Eine MECHATROLINK-II-Drehzahlachse hat ein 32-Bit Maximalsollwert.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXIS**, **S_REF**, **S_REF_OUT**, **SERVO**.

3.2.202 OV_GAIN

Typ Achsenparameter

Syntax **OV_GAIN**

Beschreibung	Der Parameter **OV_GAIN** enthält die Ausgangsdrehzahlverstärkung. Die Leistungsbeeinflussung der Ausgangsdrehzahl wird berechnet, indem die Änderung an der gemessenen Position mit dem Wert des Parameters **OV_GAIN** multipliziert wird. Der Standardwert ist 0. Ein Hinzufügen der Ausgangsdrehzahlverstärkung zu einem System entspricht mechanisch dem Hinzufügen einer Dämpfung. Sie ermöglicht ein sanfteres Ansprechen und erlaubt den Einsatz einer höheren Proportionalverstärkung als normalerweise möglich, jedoch auf Kosten eines höheren Schleppfehlers. Hohe Werte können Schwingungen verursachen und hohe Schleppfehler erzeugen. Hinweis: Um Instabilität zu vermeiden, sollte die Servoverstärkung nur dann geändert werden, wenn **SERVO** ausgeschaltet ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**D_GAIN**, **I_GAIN**, **P_GAIN**, **VFF_GAIN**.

3.2.203 P_GAIN

Typ	Achsenparameter
Syntax	**P_GAIN**
Beschreibung	Der Parameter **P_GAIN** enthält die Proportionalverstärkung. Die proportionale Leistungsbeeinflussung wird berechnet, indem der Schleppfehler mit dem Wert des Parameters **P_GAIN** multipliziert wird. Der Standardwert von **P_GAIN** für Mechatro-Speed-Achsen (**ATYPE = 41**) ist 131072. Der Standardwert für Flex Achsmodule Servo (**ATYPE = 44**) ist 1,0. Die Proportionalverstärkung stellt die Steifigkeit für das Servo-Ansprechverhalten ein. Zu hohe Werte können zu Schwingungen führen. Zu niedrige Werte führen zu großen Schleppfehlern. Hinweis: Um Instabilität zu vermeiden, sollte die Servoverstärkung nur dann geändert werden, wenn **SERVO** ausgeschaltet ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**D_GAIN**, **I_GAIN**, **OV_GAIN**, **VFF_GAIN**.

3.2.204 PI

Typ	Konstante (Nur-Lesen)
Syntax	**PI**
Beschreibung	Die Konstante **PI** gibt den numerischen Wert 3,1416 zurück.
Argumente	–
Beispiel	**Umfang = 100** **PRINT "Radius = ";Umfang/(2 * PI)**
Siehe auch	–

3.2.205 PMOVE

Typ	Task-Parameter (Nur-Lesen)
Syntax	**PMOVE**
Beschreibung	Der Parameter **PMOVE** enthält den Status des Task-Befehlsspeichers. Der Parameter gibt **TRUE** zurück, wenn die Task-Befehlsspeicher belegt sind und **FALSE**, wenn Sie leer sind. Wenn der Task einen Bewegungsbefehl ausführt, lädt sie die Bewegungsdaten in den Task-Bewegungs-Befehlsspeicher. Die Befehlsspeicher können eine Bewegungsanweisung für eine beliebige Achsengruppe speichern. **PMOVE** wird auf **TRUE** gesetzt, wenn das Laden der Befehlsspeicher abgeschlossen ist. Wenn der nächste Servo-Interrupt auftritt, lädt der Profilgenerator die Bewegung in den Befehlsspeicher für die nächste Bewegung- (**NTYPE**) der benötigten Achsen, wenn sie verfügbar sind. Wenn diese zweite Übertragung abgeschlossen ist, wird **PMOVE** auf 0 gesetzt, bis eine weitere Bewegung im Task ausgeführt wird. Jede Task hat ihren eigenen **PMOVE**-Parameter. Verwenden Sie den Modifikator **PROC**, um auf den Parameter für eine bestimmte Task zuzugreifen. Ohne **PROC** wird der aktuelle Task ausgeführt.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**NTYPE**, **PROC**.

3.2.206 POS_OFFSET

Typ	Systemparameter
Syntax	**POS_OFFSET = Wert**
Beschreibung	Für Piezo-Betrieb. Dieses Schlüsselwort erlaubt die Anwendung eines positiven Offsets auf das Ausgangs-**DAC**-Signal vom Servo-Regelkreis. Der Offset wird nach der Funktion **DAC_SCALE** angewendet. Ein Offset von 327 entspricht einem Offset von 0,1 Volt. Es wird vorgeschlagen, als Offset 65 % bis 70 % des erforderlichen Wertes anzuwenden, um eine stufige Bewegung bei offenem Regelkreis zu erreichen.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.207 POWER_UP

Typ	Systemparameter
Syntax	**POWER_UP**
Beschreibung	Mit diesem Parameter wird ermittelt, ob Programme beim Hochfahren oder Zurücksetzen der Software aus dem Flash-EPROM gelesen werden sollen (**EX**). Zwei Werte sind möglich: 0: Verwendung der Programme im batteriegestützten RAM; 1: Kopieren von Programmen aus dem Flash-EPROM des Controllers in den RAM. Mit dem Befehl **RUNTYPE** können einzelne Programme ausgewählt werden, die beim Hochfahren ausgeführt werden sollen. Hinweise: • **POWER_UP** ist immer ein unmittelbarer Befehl und kann deshalb nicht in Programme eingebunden werden. • Dieser Wert wird normalerweise durch Trajexia Tools eingestellt.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**EPROM**

3.2.208 PRINT

Typ	E/A-Befehl
Syntax	**PRINT [#n,] Ausdruck { , Ausdruck}** **? [#n,] Ausdruck { , Ausdruck}**
Beschreibung	Der Befehl **PRINT** gibt eine Zeichenserie an die seriellen Schnittstellen aus. **PRINT** kann Parameter, festgelegte ASCII-Zeichenketten und einzelne ASCII-Zeichen ausgeben. Mit **PRINT #n** kann jede beliebige Schnittstelle zur Ausgabe der Daten ausgewählt werden. Mehrere zu druckende Elemente können in die gleiche Zeile gesetzt und durch ein Komma oder Semikolon getrennt werden. Ein Komma-Trennzeichen im Druckbefehl setzt einen Tabulator zwischen die gedruckten Elemente. Bei einem Semikolon-Trennzeichen wird das nächste Element ohne Leerzeichen zwischen den Elementen gedruckt. Die Breite des Feldes, in dem eine Zahl gedruckt wird, kann durch Hinzufügen von [w,x] nach der Zahl eingestellt werden, die gedruckt werden soll. Die Spaltenbreite wird durch w und die Anzahl der Dezimalstellen durch x bestimmt. Wird nur der Parameter [x] verwendet, wird die angegebene Anzahl der Dezimalstellen in der Standardbreite gedruckt. Die Zahlen werden in dem Feld rechtsbündig dargestellt, wobei alle unbenutzten Zeichenstellen mit Leerzeichen aufgefüllt werden. Wenn die Zahl zu lang ist, wird das Feld mit Sternchen aufgefüllt, um anzuzeigen, dass nicht genug Platz zur Verfügung stand, um die Zahl darzustellen. Die maximal zulässige Feldbreite beträgt 127 Zeichen. Der Befehl „inverser Schrägstrich“\kann zum Drucken eines einzelnen ASCII-Zeichens verwendet werden.
Argumente	• **n** Das spezifizierte Ausgabegerät. Wird dieses Argument ausgelassen, wird die Schnittstelle verwendet, die durch **OUTDEVICE** spezifiziert ist. Siehe Tabelle unten. • **Ausdruck** Der Ausdruck, der gedruckt werden soll.
Beispiel	**PRINT "GROSSBUCHSTABEN und Kleinbuchstaben KOENNEN GEDRUCKT WERDEN"**
Beispiel	Bei VR(1) = 6 und **variab** = 1,5 ist die Druckausgabe wie folgt: **PRINT 123.45,VR(1)-variab** **123.4500 4.5000**

Beispiel **laenge:**
PRINT "DISTANCE = ";mpos
DISTANCE = 123.0000
In diesem Beispiel wird ein Semikolon als Trennzeichen benutzt. Dadurch springt der Cursor nicht in die nächste Spalte, wodurch der Programmierer mehr Spielraum bei der Platzierung der Druckelemente hat.

Beispiel **PRINT VR(1)[4,1];variab[6,2]**
6.0 1.50

Beispiel **parameter:**
PRINT "DISTANCE = ";mpos[0];" SPEED = ";v[2];
DISTANCE = 123 SPEED = 12.34

Beispiel **PRINT "ITEM ";gesamt" OF ";Grenzwert;CHR(13);**

Beispiel **>> PRINT HEX(15),HEX(–2)**
F FFFFA

Siehe auch **$ (HEXADEZIMALE EINGABE)**, **OUTDEVICE**.

Wert	Beschreibung
0	Programmierschnittstelle 0 (standard)
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2
5	Trajexia Tools Schnittstelle 0 Benutzerkanal 5
6	Trajexia Tools Schnittstelle 0 Benutzerkanal 6
7	Trajexia Tools Schnittstelle 0 Benutzerkanal 7

3.2.209 PROC

Typ Task-Befehl

Syntax **PROC(task_nummer)**

Beschreibung Der Modifikator **PROC** erlaubt es, einen Prozessparameter aus einem bestimmten Prozess zu lesen oder zu schreiben. Wird er ausgelassen, wird der aktuelle Task ausgeführt.

Argumente
- **task_nummer**
  Die Nummer des Tasks, auf den zugegriffen werden soll.

Beispiel **WAIT UNTIL PMOVE PROC(3) = 0**

Siehe auch –

3.2.210 PROC_STATUS

Typ Task-Parameter

Syntax **PROC_STATUS**

Beschreibung Der Parameter **PROC_STATUS** gibt den Status des Prozesses oder des Tasks zurück. Der Parameter wird zusammen mit dem Modifikator **PROC** verwendet und kann die Werte zurückgeben, die in der Tabelle unten aufgelistet sind.

Argumente –

Beispiel **WAIT UNTIL PROC_STATUS PROC(3) = 0**

Siehe auch **PROCNUMBER**, **PROC**.

Wert	Beschreibung
0	Prozess gestoppt
1	Prozess läuft
2	Prozess im Schrittbetrieb
3	Prozess angehalten

3.2.211 PROCESS

Typ	Programmbefehl
Syntax	**PROCESS**
Beschreibung	Der Befehl **PROCESS** gibt die Statusliste aller laufenden Tasks mit der zugehörigen Task-Nummer zurück.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**HALT**, **RUN**, **STOP**.

3.2.212 PROCNUMBER

Typ	Task-Parameter (Nur-Lesen)
Syntax	**PROCNUMBER**
Beschreibung	Der Parameter **PROCNUMBER** enthält die Nummer der Tasks, in dem das aktuell ausgewählte Programm läuft. **PROCNUMBER** wird oft benötigt, wenn mehrere Kopien eines Programms in verschiedenen Tasks laufen.
Argumente	–
Beispiel	**MOVE(laenge) AXIS(PROCNUMBER)**
Siehe auch	**PROC_STATUS**, **PROC**.

3.2.213 PROFIBUS

Typ	Systembefehl
Syntax	**PROFIBUS(baugruppen_nummer, 2,1,VR_start_ausgaenge,anzahl_ausgaenge, VR_start_eingaenge,anzahl_eingaenge)** **PROFIBUS(baugruppen_nummer,4,0)**
Beschreibung	Die Funktion **PROFIBUS** 2 konfiguriert den TJ1-PRT für den Datenaustausch mit der PROFIBUS-DP-Master-Baugruppe und definiert Bereiche im VR-Speicher, wo der E/A-Austausch stattfindet. Die Funktion **PROFIBUS** 4 gibt den Datenaustauschstatus des TJ1-PRT zurück. Eine Beschreibung der Bits im Datenaustausch-Statuswort finden Sie in der Tabelle.
Argumente	• **baugruppen_nummer** Spezifiziert die Baugruppennummer des TJ1-PRT im Trajexia-System. • **VR_start_ausgaenge** Die Startadresse im VR-Speicher des Controllers, an der sich die Ausgangsdaten vom PROFIBUS-DP-Master befinden. • **anzahl_ausgaenge** Die Anzahl der Ausgangsworte vom PROFIBUS-DP-Master im VR-Speicher. • **VR_Start_Eingänge** Die Startadresse im IR-Sprechers des Controllers, an der sich die Eingangsdaten für den PROFIBUS-DP-Master befinden. • **anzahl _eingaenge** Die Anzahl der Eingangsworte zum PROFIBUS-DP-Master im VR-Speicher.
Beispiel	**PROFIBUS (0,2,1,10,16,150,31)** In diesem Beispiel ist der TJ1-PRT für den Datenaustausch mit dem PROFIBUS-DP-Master konfiguriert, und zwar mit 16 Ausgangsworten (die vom Master empfangen werden) an den Positionen VR(10) bis VR(25) und 31 Eingangsworten (die an den Master gesendet werden) an den Positionen VR(150) bis VR(180).
Siehe auch	–

Bit	Wert	Beschreibung
0	0	Konfiguration des E/A-Datenaustausches fehlgeschlagen
	1	E/A-Datenaustausch erfolgreich konfiguriert
1	0	E/A-Daten nicht verfügbar
	1	E/A-Daten verfügbar
2	0	Datenaustausch in der Betriebsart OPERATE ist aktiv
	1	Datenaustausch in der Betriebsart CLEAR ist aktiv

3.2.214 PSWITCH

Typ E/A-Befehl

Syntax **PSWITCH(schalter, aktiviert [, achse, ausgangs_nummer, ausgangs_zustand, setz_position, ruecksetz_position])**

Beschreibung Der Befehl **PSWITCH** schaltet einen Ausgang ein, wenn eine vordefinierte Position erreicht ist und schaltet ihn aus, wenn eine zweite Position erreicht ist. Die Positionen werden als die gemessenen Absolutpositionen spezifiziert. Es gibt 16 Positionsschalter, von denen jeder einer beliebigen Achse zugeordnet werden kann. Jedem Schalter sind EIN- und AUS-Positionen und eine Ausgangsnummer zugeordnet.
Der Befehl kann zusammen mit 2 oder allen 7 Argumenten verwendet werden. Mit nur 2 Argumenten kann ein Schalter deaktiviert werden.
PSWITCH-Befehle werden bei jedem Servozyklus berechnet und das Ausgabeergebnis auf die Hardware angewendet. Die Ansprechzeit beträgt deshalb ungefähr eine Servozyklusperiode.
Hinweis: Einen Ausgang kann eingeschaltet bleiben, wenn er eingeschaltet war, als **PSWITCH** ausgeschaltet wurde. Der Befehl **OP** kann dazu verwendet werden, einen Ausgang wie folgt auszuschalten:
PSWITCH(2,OFF) OP(14,OFF) ' pswitch ausschalten, der OP 14 steuert
Hinweis: Die physikalischen Schalter, die bei **PSWITCH** verwendet werden, sind keine schnellen Hardware-Schalter, daher erfolgt das Umschalten über Software, was kleine Verzögerungen im Betrieb zur Folge haben kann. Ein schnelles Umschalten der Hardware ist nur bei Achsen möglich, die über den TJ1-FL02 angeschlossen sind. Verwenden Sie hierzu den Befehl **HW_PSWITCH**.

Argumente
- **schalter**
  Die Schalternummer. Bereich: [0,15].
- **aktiviert**
  Die Schalter-Aktivierung. Bereich: [ein, aus].
- **achse**
  Die Nummer der Achse, die den Positionseingang bereitstellt.
- **ausgangs_nummer**
  Der physikalische Ausgang, der eingestellt werden soll. Bereich: [8,31].
- **ausgangs_status**
  Der Zustand des Ausgangs. Bereich: [ein, aus].
- **setz_position**
  Die absolute Position in Benutzereinheiten, auf die der Ausgang eingestellt ist.
- **ruecksetz_position**
  Die absolute Position in Benutzereinheiten, auf die der Ausgang zurückgesetzt wird.

Beispiel Eine rotierende Welle hat einen über einen Nocken betätigten Schalter, dessen Stellung für Werkstücke verschiedener Größe geändert werden muss. An der Welle befindet sich auch ein Näherungsschalter, der den OT der Maschine anzeigt. Mit einem mechanischen Nocken ist die Umstellung zwischen einzelnen Arbeitsgängen zeitaufwendig. Dies kann vereinfacht werden, indem PSWITCH als Software-Nockenschalter verwendet wird. Der Näherungsschalter ist mit Eingang 7 verdrahtet, und der Ausgang ist Ausgang 11. Die Welle wird von Achse 0 gesteuert. Der Motor hat einen 900 ppr-Drehgeber. Der Ausgang muss ab 80 Einheiten eingeschaltet sein.
PSWITCH verwendet den Einheiten-Umwandlungsfaktor, damit die Positionen in einfachen Einheiten eingestellt werden können. Zuerst muss der Einheiten-Umwandlungsfaktor berechnet und eingestellt werden. Jeder Impuls am Drehgeber ergibt für den TJ1-MC__ vier zu zählende Flanken. Daher gibt es 3600 Flanken/Umdrehung oder 10 Flanken/Grad. Wenn Sie den Einheiten-Umwandlungsfaktor auf 10 einstellen, können Sie mit Grad arbeiten.
Als Nächstes müssen sie einen Wert für alle **PSWITCH**-Argumente bestimmen.
sw: Die Schalternummer kann jeder Schalter sein, der nicht in Gebrauch ist. In diesem Beispiel wird Nummer 0 benutzt.
en: Die Funktion des Schalters muss aktiviert sein; stellen Sie die Aktivierung (enable) auf 1.
achse: Die Welle wird durch Achse 0 gesteuert.
opno: Der gesteuerte Ausgang ist Ausgang 11.
opst: Der Ausgang muss auf 1 eingestellt sein.
setpos: Der Ausgang muss bei 80 Einheiten aktiviert werden.
rspos: Der Ausgang muss für einen Zeitraum von 120 Einheiten aktiviert sein.
All dies kann mit den folgenden Zeilen BASIC-Code ausgedrückt werden:

```
schalter:
  UNITS AXIS(0) = 10 ' Stellt den Einheiten-Umwandlungsfaktor ein
  REPDIST = 360
  REP_OPTION = ON
  PSWITCH(0,ON,0,11,ON,80,200)
```

Bei diesem Programm ist der Wiederholungsabstand auf 360 Grad und die Wiederholungsoption auf EIN eingestellt, sodass sich die Achsenposition zwischen 0 und 360 Grad bewegt.

Siehe auch **HW_PSWITCH**, **OP**, **UNITS**.

3.2.215 RAPIDSTOP

Typ Achsenbefehl

Syntax **RAPIDSTOP**
RS

Beschreibung Mit dem Befehl **RAPIDSTOP** wird die aktuelle Bewegung aller Achsen aus dem aktuellen Bewegungs-Befehlsspeicher abgebrochen (**MTYPE**). Bewegungen für Bewegungsbefehle mit Drehzahlprofil (**MOVE**, **MOVEABS**, **MOVEMODIFY**, **FORWARD**, **REVERSE**, **MOVECIRC** und **MHELICAL**) verzögern bis zum Stillstand mit der Verzögerungsrate, die durch den Parameter **DECEL** vorgegeben ist. Bewegungen für andere Befehle werden sofort gestoppt.
Hinweise:

- **RAPIDSTOP** bricht nur die zurzeit ausgeführten Bewegungen ab. Wenn weitere Bewegungen in den Befehlsspeichern für die nächste Bewegung oder in den Task-Befehlsspeichern (**NTYPE**) gespeichert sind, werden sie anschließend geladen.
- Während der Verzögerung der aktuellen Bewegungen werden weitere **RAPIDSTOP**-Befehle ignoriert.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **CANCEL**, **MTYPE**, **NTYPE**.

3.2.216 READ_BIT

Typ Systembefehl

Syntax **READ_BIT(bit_nummer, VR_nummer)**

Beschreibung Der Befehl **READ_BIT** gibt den Wert eines Bits in einer VR-Variablen zurück, 0 oder 1.

Argumente

- **bit_nummer**
  Die Nummer des Bits, das gelesen werden soll. Bereich: [0,23].
- **VR_nummer**
  Die Nummer der VR-Variablen, für die das Bit gelesen wird. Bereich: [0,1023].

Beispiel Kein Beispiel.

Siehe auch **CLEAR_BIT**, **SET_BIT**.

3.2.217 REG_POS

Typ	Achsenparameter (Nur-Lesen)
Syntax	**REG_POS**
Beschreibung	Der Achsenparameter **REG_POS** speichert in Benutzereinheiten die Position, an der eine Druckmarke erkannt wurde.
Argumente	–
Beispiel	**PRINT REG_POS AXIS(2)** Hierdurch wird die Druckmarkenposition für Achse 2 in Benutzereinheiten gedruckt.
Siehe auch	**AXIS**, **MARK**, **REGIST**.

3.2.218 REG_POSB

Typ	Achsenparameter (Nur-Lesen)
Syntax	**REG_POSB**
Beschreibung	Der Parameter **REG_POSB** speichert in Benutzereinheiten die Position, an der die zweite Druckmarke erkannt wurde.
Argumente	–
Beispiel	**PRINT REG_POSB AXIS(2)** Hierdurch wird die zweite Druckmarkenposition für Achse 2 in Benutzereinheiten gedruckt.
Siehe auch	**AXIS**, **MARKB**, **REGIST**.

3.2.219 REGIST

Typ	Achsenbefehl
Syntax	**REGIST(modus)**
Beschreibung	Mit dem Befehl **REGIST** wird die Druckmarkenerkennung eingerichtet. Der Befehl erfasst eine Achsenposition, wenn ein Druckmarkensignal erkannt wird. Bei einem TJ1-FL02 erfolgt die Erfassung durch die Hardware, daher beeinträchtigen Verzögerungen in der Software nicht die Genauigkeit der erfassten Position. Bei einer MECHATROLINK-II--Achse erfolgt die Erfassung über den Servotreiber. Bei Flexible Achse kann ein **REGIST**-Befehl über getrennte Druckmarkeneingänge zwei Druckmarkenpositionen erfassen. Wenn eine Druckmarke erkannt wurde, wird der Achsenparameter **MARK** auf EIN gesetzt, und die Position wird im Achsenparameter **REG_POS** gespeichert. Wird eine zweite Druckmarke erkannt, wird der Achsenparameter **MARKB** auf EIN gesetzt, und die Position wird im Achsenparameter **REG_POSB** gespeichert. Der Parameter ist nur auf Flex Achsmodule mit den **ATYPE**-Werten 43, 44 und 45 anwendbar. MECHATROLINK-II-Druckmarken können über die Drehgeber-Z-Marke oder die externen Druckmarkeneingänge EXT1, EXT2 oder EXT3 ausgeführt werden. Im Unterschied zu „Flex Achsmodulen“-Achsen kann nur eine Druckmarkenposition erfasst werden. Wenn eine Druckmarke erkannt wurde, wird der Achsenparameter **MARK** auf EIN gesetzt, und die Position wird im Achsenparameter **REG_POS** gespeichert. Der Befehl **REGIST** erlaubt das Speichern eines Druckmarkenereignisses in dem betreffenden Druckmarkeneingang. Wenn eine Druckmarke erkannt wurde, wird der Achsenparameter **MARK** auf EIN gesetzt, und die Position wird im Achsenparameter **REG_POS** gespeichert. Die Druckmarkensignale EXT1, EXT2 und EXT3 müssen mit dem Treiberparameter Pn511 den CN1-Eingängen zugeordnet sein. Beispielsweise setzt Pn511 = 654x die Anschlüsse von EXT1 auf CN1 pin44, EXT2 auf CN1 pin45 und EXT3 auf CN1 pin46. Die Tabelle unten zeigt, wie die externen Eingänge konfiguriert werden. Hinweis: Zum Konfigurieren der Druckmarkensignale EXT1, EXT2 und EXT3 werden entsprechend die Parameternummern Pn511.1, Pn511.2 und Pn511.3 verwendet. Pn511.0 wird nicht verwendet. Weitere Informationen finden Sie im Benutzerhandbuch des Servotreibers.

Druckmar-kensignal	Parameter-nummer	Parameter-wert	Beschreibung
EXT 1	Pn511.1	0 bis 3	Nicht verwendet
		4	Eingang von CN1 pin44 (Steigende Flanke)
		5	Eingang von CN1 pin45 (Steigende Flanke)
		6	Eingang von CN1 pin46 (Steigende Flanke)
		7	Signal immer AUS
		8	Signal immer EIN
		9 bis C	Nicht verwendet
		D	Eingang von CN1 pin44 (Fallende Flanke)
		E	Eingang von CN1 pin45 (Fallende Flanke)
		F	Eingang von CN1 pin46 (Fallende Flanke)
EXT 2	Pn511.2	Wie für EXT 1	Wie für EXT 1
EXT 3	Pn511.3	Wie für EXT 1	Wie für EXT 1

Die Definition eines einschließenden Erfassungsfensters ermöglicht es, dass eine Druckmarke nur innerhalb eines spezifizierten Bereiches von Achsenpositionen gültig ist. Mit dieser Funktion werden Druckmarkenereignisse ignoriert, wenn die gemessene Achsposition nicht größer ist als der Achsenparameter **OPEN_WIN** und kleiner als der Parameter **CLOSE_WIN**.
Die Definition eines ausschließenden Erfassungsfensters ermöglicht es, dass eine Druckmarke nur außerhalb eines spezifizierten Bereiches von Achsenpositionen gültig ist. Mit dieser Funktion werden Druckmarkenereignisse ignoriert, wenn die gemessene Achsposition nicht kleiner ist als der Achsenparameter **OPEN_WIN** und größer ist als der Parameter **CLOSE_WIN**.

Argumente

- **modus**

Der Modus spezifiziert Druckmarkeneingang und -ereignis sowie die Signalflanke, an der das Druckmarkenereignis auftritt. Der Modus spezifiziert auch das anzuwendende Erfassungsfenster und die Filterung.
Der Modus unterscheidet zwischen MECHATROLINK-II und Flex Achsmodulen. Die Funktionen für die einzelnen Bits im Modus sind in der Tabelle unten erklärt.

Beispiel **REGIST(4 + 1) AXIS (1)**
Mit diesem Befehl wird eine Druckmarkenerkennung aktiviert, die an der steigenden Flanke des Eingangssignals REG 0 für Achse 1 auftritt.

Beispiel **REGIST(48+64+128+512+1024) AXIS(2)**
Mit diesem Befehl wird eine zweite Druckmarkenerkennung aktiviert, die an der fallenden Flanke des Eingangssignals AUX IN mit ausschließendem Erfassungsfenster und Filterfunktion für Achse 2 auftritt.

Siehe auch **AXIS**, **MARK**, **MARKB**, **REG_POS**, **REG_POSB**, **OPEN_WIN**, **CLOSE_WIN**.

Bit	Funktion (Flex Achsmodule)
1, 0	Druckmarkenerkennung für: • 00: Z-Marke des Drehgebers • 01: Eingang REG 0 • 10: Eingang REG 1 • 11: Eingang AUX IN
2	Dieses Bit setzen, um Druckmarkenerkennung anzuwenden
3	Druckmarkenerkennung bei Signal: • 0: Steigende Flanke • 1: Fallende Flanke
5, 4	Zweite Druckmarkenerkennung für: • 00: Z-Marke des Drehgebers • 01: Eingang REG 0 • 10: Eingang REG 1 • 11: Eingang AUX IN
6	Dieses Bit setzen, um zweite Druckmarkenerkennung anzuwenden
7	Zweite Druckmarkenerkennung bei Signal: • 0: Steigende Flanke • 1: Fallende Flanke
9, 8	Erfassungsfenster-Funktionsauswahl: • 00: Kein Erfassungsfenster • 01: Einschließendes Erfassungsfenster • 10: Einschließendes Erfassungsfenster • 11: Ausschließendes Erfassungsfenster
10	Dieses Bit setzen, um Filterfunktion anzuwenden

Bit	Funktion (MECHATROLINK-II)
1, 0	Druckmarkenerkennung für: • 00: Z-Marke des Drehgebers • 01: Eingang EXT1 (CN1-Pin programmiert mit Pn511.1) • 10: Eingang EXT2 (CN1-Pin programmiert mit Pn511.2) • 11: Eingang EXT3 (CN1-Pin programmiert mit Pn511.3)
2–7	Nicht verwendet
9, 8	Erfassungsfenster-Funktionsauswahl: • 00: Kein Erfassungsfenster • 01: Einschließendes Erfassungsfenster • 10: Einschließendes Erfassungsfenster • 11: Ausschließendes Erfassungsfenster
10	Nicht verwendet

3.2.220 REMAIN

Typ Achsenparameter (Nur-Lesen)

Syntax **REMAIN**

Beschreibung Der Parameter **REMAIN** enthält die verbleibende Distanz bis zum Ende der aktuellen Bewegung. Er kann überprüft werden, um zu ermitteln, wie weit die Bewegung ausgeführt ist. **REMAIN** ist in Benutzereinheiten definiert.

Argumente –

Beispiel Um die Drehzahl 5 mm vor dem Ende einer Bewegung auf einen langsameren Wert zu ändern.

```
start:
  SPEED = 10
  MOVE(45)
  WAIT UNTIL REMAIN < 5
  SPEED = 1
  WAIT IDLE
```

Siehe auch **AXIS**, **UNITS**

3.2.221 REMOTE_ERROR

Typ Achsenparameter

Syntax **REMOTE_ERROR**

Beschreibung Gibt die Anzahl der Fehler bei einem digitalen Kommunikationsanschluss eines Treibers zurück.

Argumente –

Beispiel

```
>>PRINT REMOTE_ERROR
1.0000
```

Siehe auch –

3.2.222 RENAME

Typ Programmbefehl

Syntax **RENAME "alter_programm_name" "neuer_programm_name"**

Beschreibung Der Befehl **RENAME** ändert den Namen eines Programms im Verzeichnis des TJ1-MC__. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden.
Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. In Trajexia Tools kann der Benutzer den Befehl aus dem Menü **Programm** auswählen.

Argumente
- **alter_programm_name**
  Der aktuelle Name des Programms.
- **neuer_programm_name**
  Der neue Name des Programms.

Beispiel **RENAME "car" "auto"**

Siehe auch **COPY**, **DEL**, **NEW**.

3.2.223 REP_DIST

Typ	Achsenparameter
Syntax	**REP_DIST**
Beschreibung	Der Parameter **REP_DIST** enthält den Wiederholungsabstand, d. h. den zulässigen Bewegungsbereich für eine Achse, bevor die Sollposition (**DPOS**) und die gemessene Position (**MPOS**) korrigiert werden. **REP_DIST** ist in Benutzereinheiten definiert. Der genaue Bereich wird durch **REP_OPTION** kontrolliert. Der Parameter **REP_DIST** kann jeden positiven Wert außer 0 annehmen. Wenn die gemessene Position den Grenzwert erreicht hat, passt der TJ1-MC__ die Absolutpositionen an, ohne die aktuelle Bewegung oder den Servo-Algorithmus zu beeinträchtigen. Dabei kann die Sollposition nicht außerhalb des Bereichs liegen, weil die gemessene Position als Auslöser für die Anpassung verwendet wird. Bei jedem Ereignis (**DEFPOS**, **OFFPOS**, **MOVEABS**, **MOVEMODIFY**), das eine Position außerhalb des Bereichs definiert, wird die Endposition innerhalb des Bereichs neu definiert. Der Standardwert für alleAchsen ist 5000000.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **DPOS**, **MPOS**, **REP_OPTION**, **UNITS**.

3.2.224 REP_OPTION

Typ	Achsenparameter
Syntax	**REP_OPTION**
Beschreibung	Der Parameter **REP_OPTION** steuert die Anwendung des Achsenparameters **REP_DIST** und der Wiederholungsoption der Achsenbefehle **CAMBOX** und **MOVELINK**. Der Standardwert ist 0. Siehe Tabelle unten.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **CAMBOX**, **MOVELINK**, **REP_DIST**.

Bit	Beschreibung
0	Der Wiederholungsabstandbereich wird durch Bit 0 des Parameters **REP_OPTION** gesteuert. • Wenn das Bit 0 von **REP_OPTION** AUS ist, liegt der Bereich für die gewünschte und gemessene Position zwischen **-REP_DIST** und **REP_DIST**. • Wenn das Bit 0 von **REP_OPTION** EIN ist, liegt der Bereich für die gewünschte und gemessene Position zwischen 0 und **REP_DIST**.
1	Die automatische Wiederholungsoption der Befehle **CAMBOX** und **MOVELINK** wird durch Bit 1 des Parameters **REP_OPTION** gesteuert. Wird das Bit auf EIN gesetzt, erhält die Systemsoftware den Befehl, die automatische Wiederholungsoption zu beenden. Wenn die Systemsoftware die Option deaktiviert hat, löscht sie automatisch Bit 1 von **REP_OPTION**.

3.2.225 REPEAT..UNTIL

Typ	Programmsteuerungsbefehl
Syntax	**REPEAT** **Befehle** **UNTIL Bedingung**
Beschreibung	Die Struktur **REPEAT ... UNTIL** erlaubt es, das Programmsegment zwischen **REPEAT** und der **UNTIL**-Anweisung mehrmals zu wiederholen, bis die Bedingung **TRUE** wird. Hinweis: **REPEAT ... UNTIL**-Konstruktionen können unbegrenzt verschachtelt werden.
Argumente	• **Befehle** Jeder gültige BASIC-Befehlssatz. • **Bedingung** Jeder gültige BASIC-logische Ausdruck.
Beispiel	Ein Förderband soll 100 mm bei einer Geschwindigkeit von 1000 mm/s indizieren, 0,5 s warten und dann den Zyklus wiederholen, bis ein externes Zählersignal den Stoppbefehl gibt, indem Eingang 4 aktiviert wird. zyklus: **SPEED = 1000** **REPEAT** **MOVE(100)** **WAIT IDLE** **WA(500)** **UNTIL IN(4) = ON**
Siehe auch	**FOR..TO..STEP..NEXT**, **WHILE..WEND**.

3.2.226 RESET

Typ	Systembefehl
Syntax	**RESET**
Beschreibung	Der Befehl **RESET** setzt den Wert aller lokalen Variablen des aktuellen BASIC-Tasks auf 0.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**CLEAR**.

3.2.227 RETURN

Siehe **GOSUB..RETURN**.

3.2.228 REV_IN

Typ	Achsenparameter
Syntax	**REV_IN**
Beschreibung	Der Parameter **REV_IN** enthält die Nummer des Eingangs, der als Rückwärts-Endlage verwendet werden soll. Die Nummer kann eine Zahl von 0 bis 7 und 19 sein. Der gültige Eingangsbereich ist 0 bis 31. Die Werte 0 bis 15 stellen physikalisch die Eingänge des TJ1-MC__ E/A-Steckverbinders dar und werden für alle Achsen gemeinsam benutzt. Die Werte 16 bis 27 stellen Software-Eingänge dar, die beliebig in Programmen und Befehlen verwendet werden können, z. B. IN und OP. Sie gelten ebenfalls für alle Achsen. Die Werte 28 bis 31 werden direkt den am CN1-Steckverbinder vorhandenen Treiber-Eingängen zugewiesen und sind für jede Achse eindeutig. Welche Treiber-Eingänge den Eingängen 28 bis 31 zugewiesen werden, hängt von der Servotreiber-Einstellung in Parameter Pn81E ab. Die empfohlene Einstellung ist Pn81E = 0 x 4321, mit folgender Zuordnung. Wenn eine Eingangsnummer gesetzt und der Grenzwert erreicht ist, wird jede Rückwärtsbewegung dieser Achse gestoppt. Bit 5 des Achsenparameters **AXISSTATUS** wird ebenfalls gesetzt. Hinweis: Dieser Eingang ist in aktiviertem Zustand null.
Sigma II	• Eingang 28: CN1-40 • Eingang 29: CN1-41 • Eingang 30: CN1-42 • Eingang 31: CN1-43
Sigma III	• Eingang 28: CN1-13 • Eingang 29: CN1-7 • Eingang 30: CN1-8 • Eingang 31: CN1-9
Junma	• Eingang 26: CN1-2 • Eingang 27: CN1-1
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **AXISSTATUS**, **FWD_IN**.

3.2.229 REV_JOG

Typ	Achsenparameter
Syntax	**REV_JOG**
Beschreibung	Der Parameter **REV_JOG** enthält die Nummer des Eingangs, der als Jog-Rückwärtseingang verwendet werden soll. Die Nummer kann eine Zahl zwischen 0 und 7 sein. Standardmäßig ist der Parameter auf –1 eingestellt, und es ist kein Eingang ausgewählt. Hinweis: Dieser Eingang ist in aktiviertem Zustand null.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **FAST_JOG**, **FWD_JOG**, **JOGSPEED**, **UNITS**.

3.2.230 REVERSE

Typ	Achsenbefehl
Syntax	**REVERSE** **RE**
Beschreibung	Der Befehl **REVERSE** bewegt eine Achse kontinuierlich mit der Drehzahl rückwärts, die im Parameter **SPEED** eingestellt ist. Die Beschleunigungsrate ist im Achsenparameter **ACCEL** definiert. **REVERSE** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren Hinweis: Die Rückwärtsbewegung kann mit dem Befehl **CANCEL** oder **RAPIDSTOP** oder durch Erreichen des Rückwärts-Grenzwertes, der Sperre oder der Nullpunkt-Rückkehr gestoppt werden.
Argumente	–
Beispiel	**zurueck:** **REVERSE** **WAIT UNTIL IN(0) = ON ' Warte auf Stoppsignal** **CANCEL**
Siehe auch	**AXIS**, **CANCEL**, **FORWARD**, **RAPIDSTOP**.

3.2.231 RS_LIMIT

Typ	Achsenparameter
Syntax	**RS_LIMIT** **RSLIMIT**
Beschreibung	Der Achsenparameter **RS_LIMIT** enthält die Absolutposition des Rückwärts-Softwaregrenzwertes in Benutzereinheiten. Ein Softwaregrenzwert für die Rückwärtsbewegung kann aus dem Programm heraus gesetzt werden, um den Arbeitsbereich der Maschine zu steuern. Wenn der Grenzwert erreicht ist, verzögert der TJ1-MC__ auf 0, und die Bewegung wird abgebrochen. Bit 10 im Achsenparameter **AXISSTATUS** wird eingeschaltet, wenn die Achsenposition kleiner ist als **RS_LIMIT** bzw. unter dem Parameterwert liegt.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**, **FS_LIMIT**, **UNITS**.

3.2.232 RUN

Typ	Programmbefehl
Syntax	**RUN ["programm_name" [, task_nummer]]**
Beschreibung	Der Befehl **RUN** führt das Programm im TJ1-MC__ aus, das mit **Programm_name** spezifiziert ist. Bei Spezifizierung des Programmnamens führt **RUN** das aktuell ausgewählte Programm aus. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. Die Task-Nummer spezifiziert die Nummer des Tasks, in dem das Programm ausgeführt wird. Wird die Task-Nummer ausgelassen, wird das Programm im höchsten verfügbaren Task ausgeführt. **RUN** kann in ein Programm integriert werden, um ein anderes Programm zu starten. Hinweis: Das Programm wird ausgeführt, bis eines der folgenden Ereignisse eintritt: • Es sind keine Programmzeilen mehr zur Ausführung vorhanden. • **HALT** wird in die Befehlszeile eingegeben, um alle Programme zu stoppen. • **STOP** wird in die Befehlszeile eingegeben, um ein einzelnes Programm zu stoppen. • Im Programm ist ein **STOP**-Befehl enthalten. • Es tritt ein Laufzeitfehler auf.

Argumente
- **programm_name**
  Jeder gültige Programmname.
- **task_nummer**
  Jede gültige Task-Nummer. Bereich: [1,14].

Beispiel **>> SELECT "PROGRAM"**
PROGRAM ausgewählt
>> RUN
In diesem Beispiel wird das aktuell ausgewählte Programm ausgeführt.

Beispiel **RUN "wurst"**
In diesem Beispiel wird das Programm mit dem Namen **wurst**ausgeführt.

Beispiel **RUN "wurst",3**
In diesem Beispiel wird das Programm mit dem Namen **wurst** in Task 3 ausgeführt.

Siehe auch **HALT**, **STOP**.

3.2.233 RUN_ERROR

Typ Task-Parameter (Nur-Lesen)

Syntax **RUN_ERROR**

Beschreibung Der Parameter **RUN_ERROR** enthält die Nummer des letzten BASIC-Laufzeitfehlers, der im spezifizierten Task aufgetreten ist.
Jeder Task hat seinen eigenen **RUN_ERROR**-Parameter. Verwenden Sie den Modifikator **PROC**, um auf den Parameter für einen bestimmten Task zuzugreifen. Ohne **PROC** wird der aktuelle Task ausgeführt.

Argumente –

Beispiel **>> PRINT RUN_ERROR PROC(5)**
9.0000

Siehe auch **BASICERROR**, **ERROR_LINE**, **PROC**.

3.2.234 RUNTYPE

Typ Programmbefehl

Syntax **RUNTYPE "programm_name", auto_run [, task_nummer]**

Beschreibung Der Befehl **RUNTYPE** legt fest, ob das Programm, das durch **programm_name** spezifiziert ist, automatisch beim Hochfahren ausgeführt wird und in welchem Task es ausgeführt werden soll. Die Task-Nummer ist optional; wird sie ausgelassen, wird das Programm im höchsten verfügbaren Task ausgeführt.
Der aktuelle **RUNTYPE**-Status der einzelnen Programme wird angezeigt, wenn der Befehl **DIR** eingegeben wird. Wenn ein Programm Kompilierungsfehler aufweist, werden keine Programme beim Hochfahren gestartet. Um einen **RUNTYPE**-Befehl über Trajexia Tools einzugeben, wählen Sie **Autostart-Modus** aus dem Menü **Programm**.
Hinweis: Die Ausführung des Befehls **EPROM** ist erforderlich, um die neuen **RUNTYPE**-Einstellungen im Flash-Speicher zu speichern. Andernfalls gehen die neuen Einstellungen verloren, wenn die Spannungsversorgung ausgeschaltet wird.

Argumente
- **programm_name**
  Der Name des Programms, für das **RUNTYPE** eingestellt wird.
- **auto_run**
  0 = Manuelle Ausführung auf Befehl; 1 = Automatische Ausführung beim Hochfahren. Alle Werte, die nicht 0 sind, werden als 1 definiert.
- **task_nummer**
  Die Nummer des Tasks, in dem das Programm ausgeführt wird. Bereich: [1, 14].

Beispiel **>> RUNTYPE progname,1,3**
Diese Zeile gibt an, dass das Programm **progname** automatisch beim Hochfahren in Task 3 gestartet werden soll.

Beispiel **>> RUNTYPE progname,0**
Diese Zeile gibt an, dass das Programm **progname** manuell gestartet wird.

Siehe auch **AUTORUN**, **EPROM**, **EX**.

3.2.235 S_REF

Typ	Achsenparameter
Syntax	**DAC** **S_REF**
Beschreibung	Dieser Parameter enthält den Wert für den Drehzahlsollwert, der direkt auf den Servotreiber angewendet wird, wenn die Achse bei offenem Regelkreis betrieben wird (**SERVO = OFF**). Der Bereich dieses Parameters ist durch die Anzahl der verfügbaren Bits definiert. Bei MECHATROLINK-Achsen nimmt **S_REF** 32 Bits an, daher ist der verfügbare Bereich [–2147483648, 2147483648], was einem Spannungsbereich von [–10 V, 10 V] entspricht. Bei „Flex Achsmodul"-Achsen nimmt **S_REF** 16 Bits an, daher ist der verfügbare Bereich [–32768, 32767], was einem Spannungsbereich von [–10 V, 10 V] entspricht. Diese Bereiche können mit dem Parameter **OUTLIMIT** eingeschränkt werden. Der tatsächliche Drehzahlsollwert hängt vom Servomotor ab. Um den Drehzahlsollwert in U/min zu ermitteln, multiplizieren Sie den Parameterwert mit dem Parameterwert von **S_RATE**. Der aktuell auf den Treiber angewandte Wert kann mit dem Achsenparameter **S_REF_OUT** abgerufen werden.
Argumente	–
Beispiel	**WDOG = ON** **SERVO = OFF** **quadrat:** **S_REF AXIS(0) = 2000** **WA(250)** **S_REF AXIS(0) = –2000** **WA(250)** **GOTO Quadrat** Diese Zeilen können dazu verwendet werden, mit einem Zeitraum von ca. 500 ms an Achse 0 eine quadratische Welle mit positiver und negativer Bewegung zu erzwingen.
Siehe auch	**AXIS**, **S_REF_OUT**, **OUTLIMIT**, **SERVO**.

3.2.236 S_REF_OUT

Typ	Achsenparameter (Nur-Lesen)
Syntax	**DAC_OUT** **S_REF_OUT**
Beschreibung	Der Parameter **S_REF_OUT** enthält den Wert für den Drehzahlsollwert, der sowohl bei offenem als auch bei geschlossenem Regelkreis auf den Servotreiber angewendet wird. Bei geschlossenem Regelkreis (**SERVO = ON**) gibt der Bewegungssteuerungs-Algorithmus ein Drehzahlsollwertsignal aus, das durch die Einstellungen für die Regelungsverstärkung und den Schleppfehler bestimmt wird. Die Position des Servomotors wird über Achsenbefehle bestimmt. Bei offenem Regelkreis (**SERVO = OFF**) wird das Drehzahlsollwertsignal durch den Achsenparameter **S_REF** bestimmt. Der tatsächliche Drehzahlsollwert hängt vom Servomotor ab. Um den Drehzahlsollwert in U/min zu ermitteln, multiplizieren Sie den Parameterwert **S_REF** mit dem Parameterwert von **S_RATE**.
Argumente	–
Beispiel	**>> PRINT S_REF_OUT AXIS(0)** **288.0000**
Siehe auch	**AXIS**, **S_REF**, **OUTLIMIT**, **SERVO**.

3.2.237 SCOPE

Typ Systembefehl

Syntax **SCOPE(steuerung, periode, tabellen_anfang, tabellen_ende, P0 [, P1 [, P2 [, P3]]])**

Beschreibung Der Befehl **SCOPE** programmiert das System so, dass bei jedem Abtastintervall automatisch bis zu 4 Parameter gespeichert werden.
Die Datenspeicherung beginnt, sobald der Befehl **TRIGGER** ausgeführt ist.
Das Abtastintervall kann jedes beliebige Vielfache der Servo Zykluszeit sein.
Die Parameter werden im TABLE-Array gespeichert und können anschließend zu einem Computer zurückgelesen und auf dem Trajexia Tools-Oszilloskop angezeigt oder zur weiteren Analyse mit der Option **Tabellendatei anlegen** im Menü **Datei** in eine Datei geschrieben werden.
Die aktuelle TABLE-Position für den ersten Parameter, der von **SCOPE** geschrieben wird, kann aus dem Parameter **SCOPE_POS** ausgelesen werden.
Hinweise:
1. Trajexia Tools verwendet den Befehl **SCOPE**, wenn die Oszilloskop-Funktion ausgeführt wird.
2. Um die Berechnungszeit zum Schreiben der Echtzeitdaten zu minimieren, schreibt der Befehl **SCOPE** Rohdaten in das TABLE-Array. Beispiel:
a) Die Parameter werden in Drehgeber-Flanken (pro Sekunde) geschrieben und können daher nicht mit dem Umwandlungsfaktor **UNITS** umgewandelt werden.
b) Der Parameter **MSPEED** wird als Änderung in Drehgeber-Flanken pro Servo Zykluszeit geschrieben.
3. Die Befehle **CAM**, **CAMBOX** und **SCOPE** benutzen alle die gleiche TABLE als Datenbereich.

Argumente
- **steuerung**
  Kann auf EIN oder AUS gesetzt werden, um die Ausführung von **SCOPE** zu steuern. Wird der Status auf EIN gesetzt, ist der Befehl **SCOPE** zur Ausführung bereit, sobald der Befehl **TRIGGER** ausgeführt wird.
- **periode**
  Die Anzahl der Servozyklen zwischen den Datenerfassungen.
- **tabellen_anfang**
  Die Adresse des ersten Elements im TABLE-Array, bei dem die Datenspeicherung beginnen soll.
- **tabellen_ende**
  Die Adresse des letzten Elements im TABLE-Array, das verwendet werden soll.
- **P0**
  Der erste zu speichernde Parameter.
- **P1**
  Ein optionaler zweiter zu speichernder Parameter.
- **P2**
  Ein optionaler dritter zu speichernder Parameter.
- **P3**
  Ein optionaler vierter zu speichernder Parameter.

Beispiel **SCOPE(ON,10,0,1000,MPOS AXIS(1),DPOS AXIS(1))**
In diesem Beispiel ist die Funktion **SCOPE** so programmiert, dass sie den Parameter **MPOS** für Achse 1 und den Parameter **DPOS** für Achse 1 alle 10 Servo-Zyklen speichert. Der Parameter **MPOS** wird an den TABLE-Speicheradressen 0 bis 499 gespeichert, der Parameter **DPOS**an den TABLE-Speicheradressen 500 bis 999. Die Funktion **SCOPE** kehrt zurück und beginnt mit dem Speichern am Anfang, bis sie gestoppt wird.
Die Abtastung startet erst, wenn der Befehl **TRIGGER** ausgeführt wird.

Beispiel **SCOPE(OFF)**
Mit der obigen Zeile wird die Funktion SCOPE ausgeschaltet.

Siehe auch **SCOPE_POS**, **TABLE**, **TRIGGER**.

3.2.238 SCOPE_POS

Typ	Systemparameter (Nur-Lesen)
Syntax	**SCOPE_POS**
Beschreibung	Der Parameter **SCOPE_POS** enthält die aktuelle TABLE-Position, an der der Befehl **SCOPE** aktuell den ersten Parameter speichert.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**SCOPE**.

3.2.239 SELECT

Typ	Programmbefehl
Syntax	**SELECT "programm_name"**
Beschreibung	Der Befehl **SELECT** spezifiziert das aktuelle Programm zum Bearbeiten, Ausführen, Auflisten usw. **SELECT** erzeugt ein neues Programm, wenn der Name nicht existiert. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. Wenn ein Programm ausgewählt ist, können die Befehle **COMPILE**, **DEL**, **EDIT**, **LIST**, **NEW**, **RUN**, **STEPLINE**, **STOP** und **TROFF** auf das aktuell ausgewählte Programm angewendet werden, falls nicht ein Programm in der Befehlszeile spezifiziert wird. Wird ein anderes Programm ausgewählt, wird das zuvor ausgewählte Programm kompiliert. Das ausgewählte Programm kann nicht geändert werden, wenn ein Programm ausgeführt wird. Hinweis: Dieser Befehl ist für ein Offline- (VT100-) Terminal implementiert. Trajexia Tools wählt Programme automatisch aus, wenn Sie auf den entsprechenden Eintrag in der Liste auf der Bedienkonsole klicken.
Argumente	–
Beispiel	**>> SELECT "PROGRAM"** **PROGRAM ausgewählt** **>> RUN**
Siehe auch	**COMPILE**, **DEL**, **EDIT**, **LIST**, **NEW**, **RUN**, **STEPLINE**, **STOP**, **TROFF**.

3.2.240 SERVO

Typ	Achsenparameter
Syntax	**SERVO**
Beschreibung	Der Parameter **SERVO** ermittelt, ob die Basisachse bei geschlossenem (**SERVO = ON**) oder bei offenem Lage-Regelkreis (**SERVO = OFF**) betrieben wird. Bei geschlossenem Regelkreis gibt der Bewegungssteuerungs-Algorithmus ein Drehzahlsollwertsignal aus, das durch die Einstellungen für die Regelungsverstärkung und den Schleppfehler bestimmt wird. Die Position des Servomotors wird über Achsenbefehle bestimmt. Bei offenem Regelkreis wird das Drehzahlsollwertsignal ausschließlich durch den Achsenparameter **S_REF** bestimmt.
Argumente	–
Beispiel	**SERVO AXIS(0) = ON ' Lageregelung Achse 0 ein** **SERVO AXIS(1) = OFF ' Achse 1 wird bei offenem Lage-Regelkreis betrieben**
Siehe auch	**AXIS**, **FE_LIMIT**, **S_REF**, **S_REF_OUT**, **WDOG**.

3.2.241 SERVO_PERIOD

Typ	Systemparameter
Syntax	**SERVO_PERIOD**
Beschreibung	Mit dem Parameter **SERVO_PERIOD** wird die Servozyklus-Periode des TJ1-MC__ eingestellt. Das Timing der Ausführung der Programm-Tasks und die Auffrischung der Steuerdaten und E/A der Baugruppe hängen alle von dieser Einstellung ab. Der Parameter ist in Mikrosekunden definiert. Der TJ1-MC__ kann auf einen Servozyklus von 0,5, 1,0 oder 2,0 ms eingestellt werden. Siehe Tabelle unten.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**DRIVE_RESET**.

Wert	Beschreibung
500	0,5 ms
1000	1,0 ms
2000	2,0 ms

Achtung
Wenn der Parameter eingestellt worden ist, muss das komplette System neu gestartet oder eine Software-Rücksetzung (mit dem Befehl **EX**) durchgeführt werden. Andernfalls besteht die Gefahr von unkontrolliertem Verhalten.

3.2.242 SET_BIT

Typ Systembefehl

Syntax **SET_BIT(bit_nummer, vr_nummer)**

Beschreibung Der Befehl **SET_BIT** setzt ein Bit in einer VR-Variablen auf eins. Andere Bits in der Variablen behalten ihre Werte.

Argumente
- **bit_nummer**
  Die Nummer des Bits, das gesetzt werden soll. Bereich: [0,23].
- **vr_nummer**
  Die Nummer der VR-Variablen, für die das Bit gesetzt wird. Bereich: [0,1023].

Beispiel Kein Beispiel.

Siehe auch **CLEAR_BIT**, **READ_BIT**, **VR**.

3.2.243 SETCOM

Typ Kommunikationsbefehl

Syntax **SETCOM(baud_rate, daten_bits, stopp_bits, paritaet, port_nummer, modus)**

Beschreibung Der Befehl **SETCOM** stellt die serielle Kommunikation für die seriellen Schnittstellen ein. Der Befehl aktiviert die Host-Link-Protokolle oder definiert die Mehrzweck-Kommunikation.
Standardmäßig sind bei den seriellen Schnittstellen 9600 Baud, 7 Datenbits, 2 Stoppbits, gerade Parität und XON/XOFF für die Mehrzweck-Kommunikation aktiviert. Diese Standardeinstellungen werden beim Hochfahren wiederhergestellt.

Argumente
- **baud_rate**
  1200, 2400, 4800, 9600, 19200, 38400
- **daten_bits**
  7, 8
- **stopp_bits**
  1, 2
- **paritaet**
  0 = keine; 1 = ungerade; 2 = gerade.
- **port_nummer**
  Siehe erste Tabelle unten.
- **modus**
  Wählen Sie für die seriellen Schnittstellen 1 und 2 eine der in der zweiten Tabelle unten aufgelisteten Betriebsarten aus.

Beispiel Kein Beispiel.

Siehe auch –

port_nummer Wert	Beschreibung
1	Serielle RS-232C-Schnittstelle 1
2	Serielle RS-422A/485-Schnittstelle 2

Betriebsart	Beschreibung
0	Mehrzweck-Kommunikation (kein XON/XOFF-Mechanismus)
5	Host-Link Slave-Protokoll
6	Host-Link Master-Protokoll

3.2.244 SGN

Typ Mathematische Funktion

Syntax **SGN(ausdruck)**

Beschreibung Die Funktion **SGN** gibt das Vorzeichen einer Zahl zurück. Sie gibt den Wert 1 für positive Werte (einschließlich 0) und den Wert –1 für negative Werte zurück.

Argumente
- **ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel
```
>> PRINT SGN(–1,2)
–1.0000
```

Siehe auch –

3.2.245 SIN

Typ Mathematische Funktion

Syntax **SIN(ausdruck)**

Beschreibung Die Funktion **SIN** gibt den Sinus des Ausdrucks zurück. Die Eingabewerte sind in rad und können jeden beliebigen Wert annehmen. Der Ergebniswert liegt im Bereich zwischen –1 und 1.

Argumente
- **ausdruck**
  Jeder gültige BASIC-Ausdruck.

Beispiel
```
>> PRINT SIN(PI/2)
1.0000
```

Siehe auch –

3.2.246 SLOT

Typ Slot-Modifikator

Syntax **SLOT**

Beschreibung Dieser Modifikator spezifiziert die Baugruppennummer für einen Parameter, z. B. **COMMSTYPE**. Die Trajexia-Baugruppennummern sind 0 bis 6.

Argumente –

Beispiel Kein Beispiel.

Siehe auch –

3.2.247 SPEED

Typ Achsenparameter

Syntax **SPEED**

Beschreibung Der Parameter **SPEED** enthält die Solldrehzahl in Benutzereinheiten/s. Er kann jeden positiven Wert einschließlich 0 annehmen. Die Solldrehzahl ist die maximale Drehzahl für die Bewegungsbefehle mit Drehzahlprofil.

Argumente –

Beispiel
```
SPEED = 1000
PRINT "Drehzahl einstellen = ";SPEED
```

Siehe auch **ACCEL**, **AXIS**, **DATUM**, **DECEL**, **FORWARD**, **MOVE**, **MOVEABS**, **MOVECIRC**, **MOVEMODIFY**, **REVERSE**, **UNITS**.

3.2.248 SQR

Typ	Mathematische Funktion
Syntax	**SQR(ausdruck)**
Beschreibung	Die Funktion **SQR** gibt die Quadratswurzel des Ausdrucks zurück. Der Ausdruck muss einen positiven Wert (einschließlich 0) haben.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT SQR(4)** **2.0000**
Siehe auch	–

3.2.249 SRAMP

Typ	Achsenparameter
Syntax	**SRAMP**
Beschreibung	Der Parameter **SRAMP** enthält den S-Kurvenfaktor. Der S-Kurvenfaktor steuert den Rundungsbetrag, der auf Trapezprofile angewendet wird. Bei dem Wert 0 erfolgt keine Rundung. Bei dem Wert 10 erfolgt maximale Rundung. Der Standardwert des Parameters ist 0. **SRAMP** wird auf die Befehle **FORWARD**, **MOVE**, **MOVEABS**, **MOVECIRC**, **MHELICAL** und **REVERSE** angewendet. Hinweise: • Die Verwendung von S-Kurven erhöht die Zeit, die für den vollständigen Ablauf der Bewegung erforderlich ist. • Der S-Kurvenfaktor darf nicht geändert werden, während eine Bewegung stattfindet.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**AXIS**.

3.2.250 STEP

Siehe **FOR..TO..STEP..NEXT**.

3.2.251 STEP_RATIO

Typ	Achsenbefehl
Syntax	**STEP_RATIO(ausgangs_zaehler, dpos_zaehler)**
Beschreibung	Dieser Befehl stellt ein Verhältnis für den Achsen-Schrittmotorausgang ein. In jedem Servozyklus wird die Anzahl der Schritte mit der Funktion **STEP_RATIO** berechnet, bevor sie an den Schritt-Impulsausgang weitergeleitet wird. Impulszählerausgang = (zaehler)/(nenner) * **MPOS**. **STEP_RATIO** wirkt sich auf **MOVECIRC** und **CAMBOX** aus. Hinweise: • Die Funktion **STEP_RATIO** wird berechnet, bevor an der Schrittmotor-Achse durch 16 geteilt wird. • Große Verhältnisse sollten vermieden werden, da sie entweder einen Verlust der Auflösung oder eine erheblich geringere Stetigkeit im Bewegungsablauf zur Folge haben. Der tatsächliche physikalische Schrittgröße x 16 ist die GRUND-Auflösung der Achse; der Einsatz dieses Befehls kann die Möglichkeit des Motion-Controllers einschränken, alle Positionen genau zu erreichen. • **STEP_RATIO** ersetzt nicht **UNITS**. Verwenden Sie **STEP_RATIO** nicht, um den x16-Faktor an der Schrittmotor-Achse zu entfernen, da dies zu einer schlechten Schrittfrequenz-Steuerung führt.
Argumente	• **nenner** Eine Integer-Zahl zwischen 0 und 16777215, die dazu verwendet wird, den Nenner in der obigen Gleichung zu definieren. • **zaehler** Eine Integer-Zahl zwischen 0 und 16777215, die dazu verwendet wird, den Zähler in der obigen Gleichung zu definieren.

Beispiel Zwei Achsen werden als X und Y eingestellt, aber die Schritte der Achsen pro mm sind nicht gleich. Interpolierte Bewegungen erfordern identische UNITS-Werte bei beiden Achsen, um die Weggeschwindigkeit konstant zu halten und um zu gewährleisten, dass MOVECIRC richtig funktioniert. Die Achse mit der niedrigeren Auflösung wird so geändert, dass sie mit der Achse mit der höheren Schrittauflösung übereinstimmt, um die größtmögliche Genauigkeit für beide Achsen zu erreichen.

```
' Achse 0: 500 Zähler pro mm (31,25 Schritte pro mm)
' Achse 1: 800 Zähler pro mm (50,00 Schritte pro mm)
BASE(0)
 STEP_RATIO(500,800)
 UNITS = 800
 BASE(1)
 UNITS = 800
```

Siehe auch –

3.2.252 STEPLINE

Typ Programmbefehl

Syntax **STEPLINE ["programm_name" [, task_nummer]]**

Beschreibung Der Befehl **STEPLINE** führt eine Zeile (d. h. „Schritte“) in dem Programm aus, das durch **programm_name** spezifiziert ist. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. Wird **STEPLINE** ohne Programmname in der Befehlszeile ausgeführt, wird das aktuell ausgewählte Programm schrittweise ausgeführt. Wird **STEPLINE** ohne Programmname in einem Programm ausgeführt, wird dieses Programm schrittweise ausgeführt. Wenn das Programm spezifiziert ist, werden alle Ereignisse dieses Programms schrittweise ausgeführt. Ein neuer Task wird gestartet, wenn es keine Kopie des laufenden Programms gibt. Wenn der Task ebenfalls spezifiziert ist, wird nur die Kopie des Programms, das im spezifizierten Task läuft, schrittweise ausgeführt. Wenn es keine Kopie des Programms gibt, das im spezifizierten Task läuft, wird ein Programm in diesem Task gestartet.

Argumente
- **programm_name**
  Der Name des Programms, das schrittweise ausgeführt werden soll.
- **task_nummer**
  Die Nummer des Tasks in dem das Programm schrittweise ausgeführt werden soll. Bereich: [1,14].

Beispiel **>> STEPLINE "foerderband"**

Beispiel **>> STEPLINE "mathe",2**

Siehe auch **RUN**, **SELECT**, **STOP**, **TROFF**, **TRON**.

3.2.253 STOP

Typ Programmbefehl

Syntax **STOP ["programm_name" [, task_nummer]**

Beschreibung Der Befehl **STOP** stoppt die Ausführung des Programms, das mit **programm_name** spezifiziert ist. Wird der Programmname ausgelassen, wird das aktuell ausgewählte Programm gestoppt. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden.
Im Fall von mehreren Ausführungen eines einzelnen Programms in verschiedenen Tasks kann die **task_nummer** verwendet werden, um den Task zu spezifizieren, der gestoppt werden soll.

Argumente
- **programm_name**
  Der Name des Programms, das gestoppt werden soll.
- **task_nummer**
  Die Nummer des Tasks in dem das Programm läuft, das gestoppt werden soll. Bereich: [1,14].

Beispiel **>> STOP progname**

Beispiel Die Zeilen ab dem Label werden in diesem Beispiel nicht ausgeführt.

```
STOP
label:
  PRINT var
  RETURN
```

Siehe auch **HALT**, **RUN**, **SELECT**.

3.2.254 SYSTEM_ERROR

Typ	Systemparameter (Nur-Lesen)
Syntax	**SYSTEM_ERROR**
Beschreibung	Der Parameter **SYSTEM_ERROR** enthält die Systemfehler, die im TJ1-System seit der letzten Initialisierung aufgetreten sind. Die Bits im Parameter **SYSTEM_ERROR** sind in der Tabelle unten angegeben.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

Bit	Beschreibung
0	BASIC-Fehler
1	Batterie-Ladezustandsfehler
2–7	Reserviert für zukünftige Verwendung
8	Baugruppen-Konfigurationsfehler (Beliebige Baugruppe im System)
9	Geräte-Konfigurationsfehler (Beliebiges Gerät im System)
10–15	Reserviert für zukünftige Verwendung
16	Baugruppen-Verlustfehler (Beliebige Baugruppe im System)
17	Abschlusswiderstand nicht montiert
18	Geräte-Verlustfehler (Beliebiges Gerät im System)

3.2.255 T_REF

Typ	Achsenparameter
Syntax	**T_REF** **DAC**
Beschreibung	Der Parameter **T_REF** enthält den Wert für den Drehmomentsollwert, der auf den Servomotor angewendet wird. Der Bereich dieses Parameters ist durch die Anzahl der verfügbaren Bits definiert. Bei MECHATROLINK-Achsen nimmt **T_REF** 32 Bits an, daher ist der verfügbare Bereich [–2147483648, 2147483648], was einem Spannungsbereich von [–10 V, 10 V] entspricht. Bei „Flex Achsmodul“-Achsen nimmt **T_REF** 16 Bits an, daher ist der verfügbare Bereich [–32768, 32767], was einem Spannungsbereich von [–10 V, 10 V] entspricht. Diese Bereiche können mit dem Parameter **OUTLIMIT** eingeschränkt werden. Der tatsächliche Drehmomentsollwert hängt vom Servomotor ab.
Argumente	–
Beispiel	**T_REF AXIS(0) = 1000**
Siehe auch	**AXIS**, **S_REF**.

3.2.256 TABLE

Typ Systembefehl

Syntax **TABLE(adresse, wert {, wert})**
TABLE(adresse)

Beschreibung Der Befehl **TABLE** schreibt Daten zu und liest Daten aus dem TABLE-Array. Die maximale Länge von TABLE beträgt 64000 Elemente. Die TABLE-Werte sind Gleitkommazahlen mit Dezimalstellen. Die TABLE kann als Alternative zu Variablen auch zum Speichern von Daten verwendet werden. Der Befehl **TABLE** hat zwei Formen.

- **TABLE(adresse, wert{, wert})** schreibt eine Folge von Werten in das TABLE-Array. Der Speicheradresse des Elementes wird durch die Adresse spezifiziert. Die Folge kann eine maximale Länge von 20 Elementen haben.
- **TABLE(adresse)** gibt den TABLE-Wert an dieser Adresse zurück.

Ein Wert in der TABLE kann schreibgeschützt sein, wenn ein Wert dieser Zahl oder ein höherer Wert bereits zuvor in die TABLE geschrieben wurde. Beispielsweise wird beim Drucken von **TABLE(1001)** eine Fehlermeldung ausgegeben, wenn der höchste TABLE-Speicheradresse, der zuvor in die TABLE geschrieben wurde, Speicheradresse 1000 ist. Die Gesamtgröße von TABLE wird durch den Parameter **TSIZE** angezeigt. Bitte beachten Sie, dass dieser Wert um eins höher ist als die höchste definierte Elementadresse. Die TABLE kann durch Eingabe des Befehls **DEL "TABLE"** oder **NEW "TABLE"** in der Befehlszeile gelöscht werden.
Hinweise:

- Anwendungen wie die Befehle **CAM**, **CAMBOX** und **SCOPE** benutzen in Trajexia Tools alle die gleiche TABLE als Datenbereich. Verwenden Sie diesen Datenbereich nicht gleichzeitig zu anderen Zwecken.
- Auf die TABLE- und VR-Daten kann aus allen laufenden Tasks zugegriffen werden. Um Probleme damit zu vermeiden, dass zwei Programm-Tasks unerwartet zu einer globalen Variablen schreiben, legen Sie die Programme so aus, dass nicht zwei Programme gleichzeitig zu einer globalen Variablen schreiben können.
- Die TABLE- und VR-Daten im RAM gehen verloren, wenn die Spannungsversorgung ausgeschaltet wird.

Argumente
- **adresse**
  Der erste Adresse im TABLE Speicher zum Lesen oder Schreiben. Bereich: [0,63999]
- **wert**
  Der Wert, der an die gegebene Speicheradresse und an die folgenden Speicheradressen geschrieben werden soll.

Beispiel **TABLE(100,0,120,250,370,470,530,550)**
Die obige Zeile lädt eine interne Tabelle, wie unten gezeigt.

Beispiel Durch die folgende Zeile wird der Wert von Speicheradresse 1000 gedruckt.
>> PRINT TABLE(1000)

Siehe auch **CAM**, **CAMBOX**, **DEL**, **NEW**, **SCOPE**, **TSIZE**, **VR**.

Tabelleneintrag	Wert
100	0
101	120
102	250
103	370
104	470
105	530
106	550

3.2.257 TABLEVALUES

Typ Systembefehl

Syntax **TABLEVALUES(adresse, anzahl_von_Punkten, format)**

Beschreibung Gibt eine Liste von TABLE-Punkten zurück, die mit der spezifizierten Nummer beginnt. Zurzeit wird nur das Format „durch Kommata getrennter Text“ unterstützt.
Hinweis: **TABLEVALUES** wird hauptsächlich bei Trajexia Tools verwendet, um schnellen Zugriff auf Werte in TABLE-Datenbanken zu ermöglichen.

Argumente
- **adresse**
  Die Nummer des ersten Punktes, der zurückgegeben werden soll.
- **anzahl_von_punkten**
  Die Gesamtanzahl der Punkte, die zurückgegeben werden sollen.
- **format**
  Das Format für die Liste.

Beispiel Kein Beispiel.

Siehe auch –

3.2.258 TAN

Typ	Mathematische Funktion
Syntax	**TAN(ausdruck)**
Beschreibung	Die Funktion **TAN** gibt den Tangens des Ausdrucks zurück. Der Ausdruck wird in rad angegeben.
Argumente	• **Ausdruck** Jeder gültige BASIC-Ausdruck.
Beispiel	**>> PRINT TAN(PI/4)** **1.0000**
Siehe auch	–

3.2.259 THEN

Siehe **IF..THEN..ELSE..ENDIF**.

3.2.260 TICKS

Typ	Task-Parameter
Syntax	**TICKS**
Beschreibung	Der Parameter **TICKS** enthält den aktuellen Zähler der Task-Taktimpulse. **TICKS** ist ein 32-Bit-Zähler, der bei jedem Servozyklus dekrementiert wird. **TICKS** kann geschrieben und gelesen werden. Er kann verwendet werden, um Zykluszeiten zu messen, Zeitverzögerungen hinzuzufügen usw. Jeder Task hat seinen eigenen **TICKS**-Parameter. Verwenden Sie den Modifikator **PROC**, um auf den Parameter für einen bestimmten Task zuzugreifen. Ohne **PROC** wird der aktuelle Task ausgeführt.
Argumente	–

Beispiel	**verzoegerung:**
	TICKS = 3000
	OP(9,ON)
	Test:
	IF TICKS <= 0 THEN
	OP(9,OFF)
	ELSE
	GOTO Test
	ENDIF

Siehe auch	–

3.2.261 TIME

Typ	Systemparameter
Syntax	**TIME**
Beschreibung	Gibt die Zeit der Echtzeit-Uhr zurück. Die zurückgegebene Zeit ist die Anzahl von Sekunden seit Mitternacht 00:00:00.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	–

3.2.262 TIME$

Typ	Systembefehl
Syntax	**TIME$**
Beschreibung	Druckt die aktuelle Zeit, wie sie durch die Echtzeit-Uhr vorgegeben ist, als Zeichenkette im 24-Stunden-Format.
Argumente	–
Beispiel	**>>? TIME$** **14/39/02**
Siehe auch	–

3.2.263 TO

Siehe **FOR..TO..STEP..NEXT**.

3.2.264 TRANS_DPOS

Typ	Achsenparameter (Nur-Lesen)
Syntax	**TRANS_DPOS**
Beschreibung	Enthält die gewünschte Achsenposition am Ausgang einer Rahmentransformation. **TRANS_DPOS** ist normalerweise gleich **DPOS** bei jeder Achse. Die Rahmentransformation ist deshalb bei jeder Achse gleich 1:1. Bei einigen Maschinenkonfigurationen kann es nützlich sein, eine Rahmentransformation einzurichten, die nicht gleich 1:1 ist. Hierbei handelt es sich typischerweise um Maschinen wie Robotarme oder Maschinen mit parasitären Bewegungen der Achsen. Rahmentransformationen müssen speziell in der C-Sprache geschrieben und in den Controller heruntergeladen werden. Wenn Sie Rahmentransformationen einrichten wollen, wenden Sie sich unbedingt an OMRON.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**FRAME**.

3.2.265 TRIGGER

Typ	Systembefehl
Syntax	**TRIGGER**
Beschreibung	Der Befehl **TRIGGER** startet einen zuvor gesetzten **SCOPE**-Befehl. Hinweis: Trajexia Tools verwendet den Befehl **TRIGGER** automatisch für die Oszilloskop-Funktion.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**SCOPE**.

3.2.266 TROFF

Typ	Programmbefehl
Syntax	**TROFF ["programm_name"]**
Beschreibung	Der Befehl **TROFF** hält eine Aufzeichnung in der aktuellen Zeile an und setzt die normale Programmausführung für das Programm fort, das mit **programm_name** spezifiziert ist. Der Programmname kann auch ohne Anführungszeichen spezifiziert werden. Wird der Programmname ausgelassen, wird das ausgewählte Programm fortgesetzt.
Argumente	• **programm_name** Der Name des Programms, bei dem die Aufzeichnung angehalten werden soll.
Beispiel	**>> TROFF "zeilen"**
Siehe auch	**SELECT**, **TRON**.

3.2.267 TRON

Typ	Programmbefehl
Syntax	**TRON**
Beschreibung	Der Befehl **TRON** erzeugt einen Haltepunkt in einem Programm, der die Programmausführung bei der Zeile anhält, die auf den Befehl **TRON** folgt. Anschließend kann das Programm z. B. zeilenweise mit dem Befehl **STEPLINE** ausgeführt werden. Hinweise: • Die Programmausführung kann ohne den Befehl **STEPLINE** wieder aufgenommen werden, indem der Befehl **TROFF** ausgeführt wird. • Der Aufzeichnungsmodus kann mit einem **STOP**- oder **HALT**-Befehl gestoppt werden. • Trajexia Tools markiert Zeilen, die einen **TRON**-Befehl enthalten, in den Fenstern **Edit** und **Debug**.
Argumente	–
Beispiel	**TRON** **MOVE(0,10)** **MOVE(10,0)** **TRON** **MOVE(0,–10)** **MOVE(–10,0)**
Siehe auch	**SELECT**, **TROFF**.

3.2.268 TRUE

Typ	Konstante (Nur-Lesen)
Syntax	**TRUE**
Beschreibung	Die Konstante **TRUE** gibt den numerischen Wert –1 zurück.
Argumente	–

Beispiel	**test:**
	t = IN(0) AND IN(2)
	IF t = TRUE THEN
	PRINT "Eingaenge sind EIN"
	ENDIF

Siehe auch	–

3.2.269 TSIZE

Typ	Systemparameter (Nur-Lesen)
Syntax	**TSIZE**
Beschreibung	Der Parameter **TSIZE** gibt die Größe des TABLE-Arrays zurück, die um 1 größer ist als das aktuell als höchstes definierte TABLE-Element. **TSIZE** wird auf 0 zurückgesetzt, wenn das TABLE-Array mit dem Befehl **DEL "TABLE"** oder **NEW "TABLE"** in der Befehlszeile gelöscht wird.
Argumente	–
Beispiel	Das folgende Beispiel geht davon aus, dass keine Speicheradresse, die höher als 1000 ist, in das TABLE-Array geschrieben worden ist. **>> TABLE(1000,3400)** **>> PRINT TSIZE** **1001.0000**
Siehe auch	**DEL**, **NEW**, **TABLE**.

3.2.270 UNITS

Typ	Achsenparameter
Syntax	**UNITS**
Beschreibung	Der Parameter **UNITS** enthält den Einheiten-Umwandlungsfaktor. Der Einheiten-Umwandlungsfaktor erlaubt es dem Benutzer, eine übersichtlichere Einheit zu definieren, z. B. m, mm oder Motorumdrehungen, indem die Anzahl der Drehgeber-Flanken spezifiziert wird, die in einer Benutzereinheit enthalten sind. In diesen Benutzereinheiten sind Achsenparameter wie Drehzahl, Beschleunigung, Verzögerung und die Achsenbefehle spezifiziert. Hinweis: Der Parameter **UNITS** kann jeden beliebigen Wert ungleich null annehmen, aber es wird empfohlen, Systeme mit einer Integer-Zahl der Drehgeberimpulse pro Benutzereinheit zu entwickeln. Eine Änderung von **UNITS** wirkt sich auf alle Achsenparameter aus, die von **UNITS** abhängig sind, um die gleiche Dynamik für das System aufrechtzuerhalten.
Argumente	–
Beispiel	Eine Leitspindelanordnung hat 5 mm Steigung und einen Drehgeber mit 1000 Impulsen/Umdrehung. Die Einheiten müssen so eingestellt werden, dass die Bewegungen in mm spezifiziert werden können. Die 1000 Impulse/Umdrehung erzeugen 1000 x 4 = 4000 Flanken/Umdrehung. Eine Umdrehung entspricht 5 mm. Daher ergeben sich 4000/5 = 800 Flanken/mm. **UNITS** wird daher wie folgt eingestellt: **>> UNITS = 1000 * 4/5**
Siehe auch	**AXIS**, **ENCODER_RATIO**.

3.2.271 UNLOCK

Siehe **LOCK**.

3.2.272 UNTIL

Siehe **REPEAT..UNTIL**.

3.2.273 VERIFY

Typ	Achsenparameter
Syntax	**VERIFY**
Beschreibung	Der Achsenparameter VERIFY wird verwendet, um verschiedene Betriebsarten bei einer Schrittmotor-Drehgeberachse auszuwählen. • **VERIFY = OFF** Der Drehgeber-Zählerschaltkreis wird mit den Hardware-Signalen **STEP** und **DIRECTION** verbunden, sodass diese wie Drehgebersignale gezählt werden. Dies ist besonders für Druckmarken nützlich, da der Druckmarken-Schaltkreis so bei einer Schrittmotorachse funktionieren kann. • **VERIFY = ON** Der Drehgeberschaltkreis ist mit den externen Signalen A,B, Z verbunden. Hinweis: Wenn **VERIFY = OFF** ist, wird beim TJ1-FL02 der Drehgeber-Zählerschaltkreis so konfiguriert, dass er **STEP**- und **DIRECTION**-Signale annimmt, die mit den Drehgebereingängen A und B fest verdrahtet sind. Wenn **VERIFY = ON** ist, wird der Drehgeberschaltkreis für den normalen Quadratureingang konfiguriert. Stellen Sie sicher, dass die Drehgebereingänge 5 Volt nicht überschreiten.
Argumente	–
Beispiel	**VERIFY AXIS(3) = ON**
Siehe auch	–

3.2.274 VERSION

Typ	Systemparameter (Nur-Lesen)
Syntax	**VERSION**
Beschreibung	Der Parameter **VERSION** gibt die aktuelle Firmware-Versionsnummer des aktuell im TJ1-MC__ installierten Systems zurück.
Argumente	–
Beispiel	**>> PRINT VERSION** **1.6100**
Siehe auch	–

3.2.275 VFF_GAIN

Typ	Achsenparameter
Syntax	**VFF_GAIN**
Beschreibung	Der Achsenparameter **VFF_GAIN** enthält die Verstärkung der Drehzahlvorsteuerung. Die Leistungsbeeinflussung der Drehzahlvorsteuerung wird berechnet, indem die Änderung an der gewünschten Position mit dem Wert des Parameters **VFF_GAIN** multipliziert wird. Der Standardwert ist 0. Das Hinzufügen einer Verstärkung der Drehzahlvorsteuerung verringert den Schleppfehler während einer Bewegung, indem die Leistung proportional zur Drehzahl erhöht wird. Hinweis: Um Instabilität zu vermeiden, sollte die Servoverstärkung nur dann geändert werden, wenn **SERVO** ausgeschaltet ist.
Argumente	–
Beispiel	Kein Beispiel.
Siehe auch	**D_GAIN**, **I_GAIN**, **OV_GAIN**, **P_GAIN**.

3.2.276 VP_SPEED

Typ	Achsenparameter (Nur-Lesen)
Syntax	**VP_SPEED**
Beschreibung	Der Parameter **VP_SPEED** enthält die Drehzahl des Drehzahlprofils in Benutzereinheiten/s. Die Drehzahl des Drehzahlprofils ist eine interne Drehzahl, die mit der Profilierung der Bewegung beschleunigt und verzögert wird.
Argumente	–
Beispiel	**' Warten bis Sollwertdrehzahl erreicht ist** **MOVE(100)** **WAIT UNTIL SPEED = VP_SPEED**
Siehe auch	**AXIS**, **MSPEED**, **UNITS**.

3.2.277 VR

Typ	Systembefehl
Syntax	**VR(adresse)**
Beschreibung	Der Befehl VR liest oder schreibt den Wert einer globalen (VR) Variablen. Diese VR-Variablen speichern reelle Zahlen und können auf einfache Art als Element oder Anordnung von Elementen verwendet werden. Der TJ1-MC__ hat insgesamt 1024 VR-Variablen. Die VR-Variablen können in der BASIC-Programmierung zu verschiedenen Zwecken verwendet werden. VR-Variablen werden global von den Tasks gemeinsam benutzt und können zur Kommunikation von Tasks untereinander verwendet werden. Hinweise: • Auf die TABLE- und VR-Daten kann aus allen laufenden Tasks zugegriffen werden. Um Probleme damit zu vermeiden, dass zwei Programm-Tasks unerwartet zu einer globalen Variablen schreiben, legen Sie die Programme so aus, dass nicht zwei Programme gleichzeitig zu einer globalen Variablen schreiben können. • Die TABLE- und VR-Daten im RAM gehen verloren, wenn die Spannungsversorgung ausgeschaltet wird.
Argumente	• **adresse** Die Adresse der VR-Variablen. Bereich: [0,1023].
Beispiel	Im folgenden Beispiel wird der Wert 1,2555 in der VR-Variablen 15 gespeichert. Die lokale Variable **val** wird dazu verwendet, die globale Variable lokal zu benennen: **val = 15** **VR(val) = 1.2555**

Beispiel

Ein Portalkran hat 10 Absetzpositionen hintereinander. Jede Position kann zu jedem Zeitpunkt belegt oder frei sein. VR(101) bis VR(110) werden dazu benutzt, eine Anordnung von 10 Einsen und Nullen zu speichern, um zu melden, ob die Positionen belegt (1) oder frei (0) sind. Der Kran setzt die Ladung auf der ersten freien Position ab. Ein Teil des entsprechenden Programms würde wie folgt aussehen:

```
bewegungp:
MOVEABS(115) ' Bewegung zur ersten Absetzposition
FOR VR(0) = 101 TO 110
IF (VR(VR(0)) = 0) THEN GOSUB last
MOVE(200) ' 200 ist der Abstand zwischen den Positionen
NEXT VR(0)
PRINT "Alle Positionen sind belegt"
WAIT UNTIL IN(3) = ON
GOTO bewegungp

last: ' Setze Last auf Position und markiere Anordnung
OP(15,OFF)
VR(VR(0)) = 1
RETURN
```

Die Variablen sind durch eine Batterie gesichert, sodass das hier gegebene Programmbeispiel dazu verwendet werden könnte, den Status der Maschine zu speichern, wenn die Spannungsversorgung ausgeschaltet ist. Es ist empfehlenswert, eine Methode zum vollständigen Zurücksetzen nach einem manuellen Eingriff bereitzustellen.

Beispiel

```
schleife: ' Zuordnung von VR(65) zu VR(0) multipliziert mit der
gemessenen Position von Achse 1
VR(65) = VR(0) * MPOS AXIS(1)
 PRINT VR(65)
GOTO schleife
```

Siehe auch **CLEAR_BIT**, **READ_BIT**, **SET_BIT**, **TABLE**.

3.2.278 VRSTRING

Typ Systembefehl

Syntax **VRSTRING(vr_start)**

Beschreibung Kombiniert den Inhalt einer Anordnung von VR()-Variablen, sodass sie als Text-Zeichenfolge gedruckt werden können. Alle druckbaren Zeichen werden ausgegeben; die Zeichenkette endet beim ersten gefundenen Null-Zeichen (d. h. VR(n) enthält 0).

Argumente

- **vr_start**
  Nummer der ersten VR() in der Zeichenanordnung.

Beispiel **PRINT #5,VRSTRING(100)**

Siehe auch –

3.2.279 WA

Typ Systembefehl

Syntax **WA(zeit)**

Beschreibung Der Befehl **WA** stoppt die Programmausführung für die angegebene Anzahl an Millisekunden. Der Befehl kann nur in einem Programm benutzt werden.

Argumente

- **zeit**
  Die Anzahl an Millisekunden, für die die Programmausführung angehalten werden soll.

Beispiel Mit den folgenden Zeilen würde Ausgang 7 eingeschaltet werden, und zwar zwei Sekunden, nachdem Ausgang 1 ausgeschaltet wurde.

```
OP(1,OFF)
WA(2000)
OP(7,ON)
```

Siehe auch –

3.2.280 WAIT IDLE

Typ	Systembefehl
Syntax	**WAIT IDLE**
Beschreibung	Der Befehl **WAIT IDLE** setzt die Programmausführung aus, bis die Basisachse ihre aktuelle Bewegung und eventuelle weitere im Befehlsspeicher befindliche Bewegungen ausgeführt hat. Der Befehl kann nur in einem Programm benutzt werden. **WAIT IDLE** wirkt sich auf die Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren Hinweis: Die Ausführung von **WAIT IDLE** bedeutet, dass das Programm wartet, bis das Bewegungsprofilder Achse abgeschlossen ist.
Argumente	–
Beispiel	**MOVE(1000)** **WAIT IDLE** **PRINT "Bewegung abgeschlossen"** Die Meldung wird am Ende der Bewegung gedruckt.
Beispiel	**MOVE(1000)** **WAIT UNTIL MTYPE = 0** **PRINT "Bewegung beendet"** Meistens wird die Meldung gedruckt, BEVOR die Bewegung startet, manchmal jedoch erst, wenn die Bewegung beendet ist.
Erläuterung	Bewegungsprogramme und Bewegungsabläufe arbeiten parallel und unsynchronisiert. Es kann ein vollständiger Zyklus durchlaufen werden, bevor die Bewegung in den Befehlsspeicher geladen wird. Das Programm führt den Befehl **MOVE(1000)** aus, aber die Bewegung wird nicht vor dem Beginn des nächsten „Bewegungsablaufes“ in den Befehlsspeicher geladen; wenn bei der Überprüfung **MTYPE = 0** festgestellt wird, ist der Parameter auch **0**, weil die Bewegung NOCH NICHT BEGONNEN HAT und nicht, weil sie beendet ist.
Siehe auch	**AXIS**, **WAIT LOADED**.

Hinweis:
WAIT IDLE ist darauf ausgelegt zu warten, bis die vorherige Bewegung beendet ist; daher wird die Verzögerung berücksichtigt, die zwischen der Befehlsausführung im Programm und dem Laden des Befehls in den Bewegungs-Befehlsspeicher entsteht.

3.2.281 WAIT LOADED

Typ	Systembefehl
Syntax	**WAIT LOADED**
Beschreibung	Der Befehl **WAIT LOADED** setzt die Programmausführung aus, bis die Basisachse keine weiteren Bewegungen außer der aktuell ausgeführten Bewegung mehr im Befehlsspeicher hat. Der Befehl kann nur in einem Programm benutzt werden. Dies ist nützlich, um Ereignisse zu Beginn oder am Ende einer Bewegung zu aktivieren, wenn sich mehrere Bewegungen zusammen im Befehlsspeicher befinden. **WAIT_LOADED** wirkt sich auf die Standard-Basisachse aus (Einstellung mit **BASE**), wenn nicht **AXIS** verwendet wurde, um eine Achse temporär zu spezifizieren.
Argumente	–
Beispiel	**' Setze Ausgang 8 auf EIN bei Start von Start von MOVE(500) und auf AUS am Ende** **MOVE(800)** **MOVE(500)** **WAIT LOADED** **OP(8,ON)** **MOVE(400)** **WAIT LOADED** **OP(8,OFF)**
Siehe auch	**AXIS**, **WAIT IDLE**

3.2.282 WAIT UNTIL

Typ	Systembefehl
Syntax	**WAIT UNTIL bedingung**
Beschreibung	Der Befehl **WAIT UNTIL** wertet die Bedingung wiederholt aus, bis sie **TRUE** ist. Anschließend wird die Programmausführung wieder aufgenommen. Der Befehl kann nur in einem Programm benutzt werden.
Argumente	• **bedingung** Jeder gültige BASIC-logische Ausdruck.

Beispiel In diesem Beispiel wartet das Programm, bis die gemessene Position bei Achse 0 den Wert 150 überschreitet und startet dann eine Bewegung bei Achse 1.
WAIT UNTIL MPOS AXIS(0)>150
MOVE(100) AXIS(1)

Beispiel Die ausgewerteten Ausdrücke können beliebig komplex sein, vorausgesetzt, dass sie der BASIC-Syntax folgen, z. B.:
WAIT UNTIL DPOS AXIS(2) <= 0 OR IN(1) = ON
Die obige Zeile sagt aus, dass gewartet werden soll, bis die Sollposition von Achse 2 kleiner oder gleich 0 ist oder bis Eingang 1 eingeschaltet ist.

Siehe auch –

3.2.283 WDOG

Typ Systemparameter

Syntax **WDOG**

Beschreibung Der Parameter **WDOG** enthält den Software-Schalter, der den Servotreiber mit dem Eingangssignal **RUN** (Servo EIN) aktiviert. Der aktivierte Servotreiber steuert den Servomotors entsprechend den Drehzahl- und Drehmomentsollwerten.
WDOG kann in der Programmsteuerung, über die Befehlszeile und die Trajexia Tools-Steuerschaltfläche ein- und ausgeschaltet werden.
Der Servotreiber wird automatisch deaktiviert, wenn ein **MOTION_ERROR** auftritt. Eine Bewegungsfehler tritt auf, wenn der **AXISSTATUS**-Zustand für eine der Achsen mit der Einstellung **ERRORMASK** übereinstimmt. In diesem Fall wird der Software-Schalter (**WDOG**) ausgeschaltet, der Parameter **MOTION_ERROR** hat den Wert 1, und der Parameter **ERROR_AXIS** enthält die Nummer der ersten Achse, bei der der Fehler auftritt.
Hinweis: Der Parameter **WDOG** kann automatisch über Trajexia Tools ausgeführt werden, wenn auf die Schaltfläche **Treiber aktivieren** auf der Bedienkonsole geklickt wird.

Argumente –

Beispiel Kein Beispiel.

Siehe auch **AXISSTATUS**, **ERROR_AXIS**, **ERRORMASK**, **MOTION_ERROR**, **SERVO**.

3.2.284 WHILE..WEND

Typ Programmsteuerungsbefehl

Syntax
```
WHILE bedingung
  Befehle
WEND
```

Beschreibung Die Struktur **WHILE ... WEND** erlaubt es, das Programmsegment zwischen **WHILE** und der **WEND**-Anweisung mehrmals zu wiederholen, bis die Bedingung **FALSE** wird. In diesem Fall wird die Programmausführung nach **WEND** wieder aufgenommen.
Hinweis: **WHILE ... WEND** – Schleifen können unbegrenzt verschachtelt werden.

Argumente
- **Bedingung**
  Jeder gültige BASIC-logische Ausdruck.

Beispiel
```
WHILE IN(12) = OFF
  MOVE(200)
  WAIT IDLE
  OP(10,OFF)
  MOVE(–200)
  WAIT IDLE
  OP(10,ON)
WEND
```

Siehe auch **FOR..TO..STEP..NEXT**, **REPEAT..UNTIL**

3.2.285 XOR

Typ	Mathematische Operation
Syntax	**ausdruck1 XOR ausdruck2**
Beschreibung	Der Operator **XOR** führt eine logische **XOR**-Verknüpfung zwischen entsprechenden Bits der Integer-Teile von zwei gültigen BASIC-Ausdrücken durch. Die logische **XOR**-Verknüpfung zwischen zwei Bits ist in der Tabelle unten definiert.
Argumente	• **ausdruck1** Jeder gültige BASIC-Ausdruck. • **ausdruck2** Jeder gültige BASIC-Ausdruck.
Beispiel	**VR(0) = 10 XOR 18** Das **XOR** ist ein Bit-Operator, daher sieht der Binärcode wie folgt aus: **01010 XOR 10010 = 11000**. Das Ergebnis ist daher 24.
Siehe auch	–

Bit 1	Bit 2	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

4 Kommunikationsprotokolle

4.1 Verfügbare Schnittstellen

Die Trajexia-Baugruppen haben folgende Schnittstellen zur Kommunikation:

Baugruppe	Schnittstelle	Protokoll	Kommentar
TJ1-MC__	Ethernet	Trajexia Tools-Protokoll	Zum Debugging sowie zur Programmierung und Überwachung des Projekts mit Trajexia Tools.
		FINS-Server	Zur Kommunikation mit einem FINS-Master, z. B. SPS, MMS oder PC.
		FINS-Client	Zur Kommunikation mit einem FINS-Server, z. B. SPS oder einer anderen Trajexia-Baugruppe.
	Seriell	Host-Link-Master	Zur Kommunikation mit einem Host-Link-Slave, z. B. einer OMRON-SPS.
		Host-Link-Slave	Zur Kommunikation mit einem Host-Link-Master, normalerweise MMS.
		Benutzerdefiniert	Dieses Protokoll wird mit BASIC-Befehlen erstellt und verwaltet.
TJ1-PRT	PROFIBUS	PROFIBUS DP-Slave DP-V0	Zum Austausch von Wort-Variablen mit einem PROFIBUS-Master.
TJ1-DRT	DeviceNet	DeviceNet	Zum Austausch von Wort-Variablen mit einem DeviceNet-Master.
TJ1-ML__	MECHATROLINK	MECHATROLINK	Zur Kommunikation mit unterstützten MECHATROLINK-Slaves. Dieses Protokoll ist für den Benutzer transparent.

4.2 Ethernet

Der TJ1-MC__ hat eine standardmäßige 10/100 Mbps Ethernet-Schnittstelle. Sie können ein Patch- oder Crossover-Ethernet-Kabel verwenden, um den TJ1-MC__ an einen PC anzuschließen.
Setzen Sie zum Konfigurieren der Schnittstelle folgende Parameter:

Eigenschaft	Standardwert	Kommentar
IP-Adresse	192.168.0.250	Vergeben Sie eine IP-Adresse, die im Netzwerk eindeutig ist.
Subnetzmaske	255.255.255.0	Vergeben Sie die gleiche Subnetzmaske, die das LAN verwendet.
Gateway	0.0.0.0	Das Gateway ist notwendig, um dezentralen Zugriff von einem anderen LAN aus zu haben.

Stellen Sie sicher, dass die IP-Adresse des PCs im gleichen Bereich wie der TJ1-MC__ liegt: Wenn die IP-Adresse des TJ1-MC__ *aaa.bbb.ccc.ddd* ist, muss die IP-Adresse des PCs *aaa.bbb.ccc.xxx* sein, wobei *xxx* 000 bis 255 außer *ddd* sein kann.
Sie können die IP-Adresse des TJ1-MC__ ändern, damit sie mit der IP-Adresse Ihres PCs übereinstimmt, wenn Sie den PC über einen Netzwerk-Hub oder Switch anschließen. Wenn z. B. die IP-Adresse des PCs 192.200.185.001 ist, können Sie die IP-Adresse des TJ1-MC__ auf 192.200.185.002 einstellen.

Hinweis
Der TJ1-MC__ hat keine DHCP-Funktionalität, daher kann er einem PC keine IP-Adresse zuordnen.

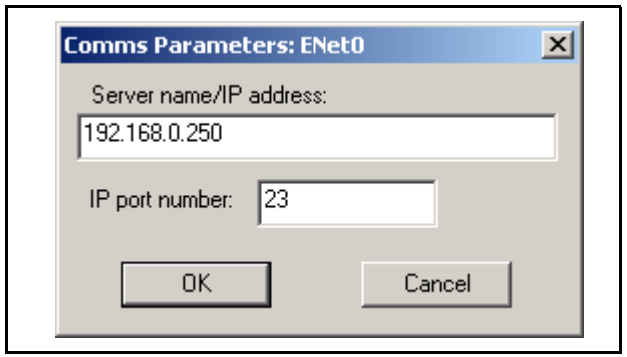
Die Subnetzmaske des TJ1-MC__ ist generisch. Sie braucht nicht mit der Subnetzmaske des PCs übereinzustimmen.
Verwenden Sie den **Ethernet**-Befehl zum Lesen oder Schreiben der Ethernet-Einstellungen. Es ist notwendig, die Baugruppen herunterzufahren und neu zu starten, damit die Änderungen übernommen werden.
Sie können die IP-Adresse des TJ1-MC__ mit der Trajexia Tools-Befehlszeile und dem Befehl **Ethernet** überprüfen: Geben Sie den Befehl **Ethernet(0,-1,0)** in die Befehlszeile ein; die IP-Adresse des TJ1-MC__ wird in der Befehlszeile angezeigt.

Hinweis
Sie müssen die Spannungsversorgung für das Trajexia-System aus- und anschließend wieder einschalten, damit die Änderung der IP-Adresse übernommen wird.

4.2.1 Kommunikation mit Trajexia direkt vom PC aus

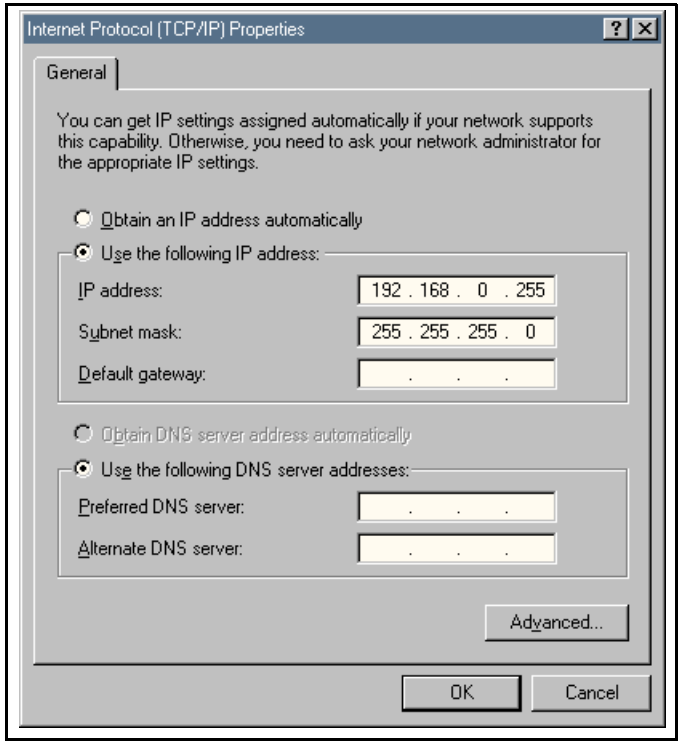
1. Verändern Sie die Ethernet-Einstellungen in Trajexia nicht.
2. Nehmen Sie die Trajexia-Tools-Einstellungen wie gezeigt vor.

Abb.1



3. Nehmen Sie die Computer-Einstellungen wie gezeigt vor.

Abb.2



4.2.2 Dezentrale Kommunikation mit Trajexia

Dieses Beispiel zeigt, wie eine Verbindung zu einem lokalen Trajexia-System von einem Computer an einem dezentralen Standort hergestellt wird. In diesem Beispiel sind die Ethernet-Einstellungen des Trajexia-Systems wie folgt:

- 10.83.50.70 ist die dem Trajexia-System zugeordnete IP-Adresse.
- 255.255.240.0 ist die lokale Subnetzmaske.
- 10.83.48.1 ist das lokale Gateway.
- Der Server ordnet den Computern automatisch eine IP-Adresse zu.

1. Stellen Sie die IP-Adresse, die Subnetzmaske und das Gateway über die Befehlszeile im Terminal-Fenster in Trajexia die folgt ein:
   Ethernet(1,-1,0,10,83,50,70)
   Ethernet(1,-1,2,255,255,240,0)
   Ethernet(1,-1,8,10,83,48,1)

2. Überprüfen Sie, ob die IP-Einstellungen des lokalen Trajexia-Systems und des dezentralen Computers wie gezeigt sind.

Nach dem Einschalten der Spannungsversorgung zeigt das Display des TJ1-MC__ abwechselnd die IP-Adresse und die Subnetzmaske an. Nachdem das Ethernet-Kabel wieder angeschlossen ist, zeigt das Display nur die IP-Adresse an.

Abb.3

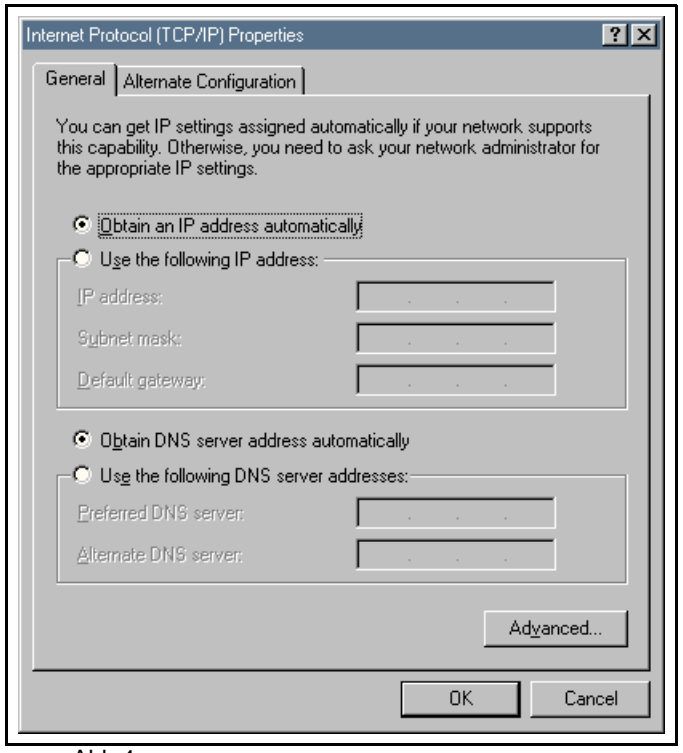
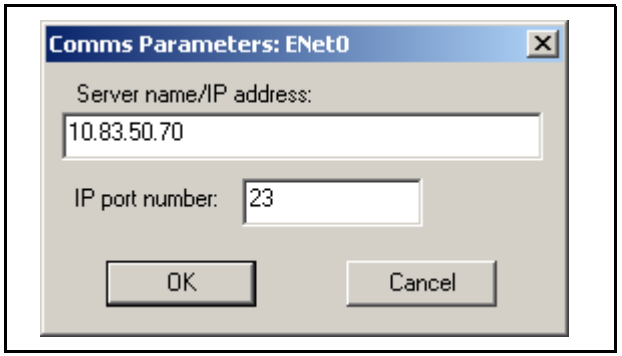


Abb.4



4.2.3 Trajexia Tools-Protokoll

Das Trajexia Tools-Protokoll dient zum Debugging sowie zur Programmierung und Überwachung TJ1-MC__.

Trajexia Tools verwendet ein Telnet-Protokoll. Standardmäßig ist dieses mit Port 23 verbunden. Wenn diese Schnittstelle nicht zur Verfügung steht, können Sie die Schnittstellennummer mit dem Befehl **Ethernet(1,-1,4,neuer_port_n)** ändern.

Im Unterschied zu den standardmäßigen Ethernet-Befehlen wird dieser Befehl unmittelbar nach seiner Ausführung übernommen. Beim Hochfahren wird die standardmäßige Schnittstelle verwendet. Deshalb muss dieser Befehl in jedes Programm eingebunden sein, das beim Hochfahren ausgeführt wird.

Das Trajexia Tools-Protokoll ist ausschließlich TCP.

4.2.4 FINS-Serverprotokoll

FINS (Factory Interface Network Service) ist ein proprietäres OMRON-Kommunikationsprotokoll. Ein Teilbereich dieses Protokolls ist in Trajexia implementiert. Ausführliche Informationen finden Sie im Referenzhandbuch für Kommunikationsbefehle (W342-E1).

Das FINS-Protokoll ermöglicht eine nahtlose Kommunikation mit anderen OMRON-Geräten, z. B. SPS, MMS und CX-Drive.
Für das FINS-Serverprotokoll sind keine Konfigurationseinstellungen nötig.

VORSICHT
Da der TJ1-MC__ mit mehreren Quellen gleichzeitig kommunizieren kann, können sich die Befehle von zwei Quellen gegenseitig stören.

Standardmäßig verwendet diese Verbindung Port 9600. Wenn diese Schnittstelle nicht zur Verfügung steht, können Sie die Schnittstelle mit dem Befehl **Ethernet(1,-1,12,neuer_port_n)** ändern.

Im Unterschied zu den standardmäßigen Ethernet-Befehlen wird dieser Befehl unmittelbar nach seiner Ausführung übernommen. Beim Hochfahren wird die standardmäßige Schnittstelle verwendet. Deshalb muss dieser Befehl in jedes Programm eingebunden sein, das beim Hochfahren ausgeführt wird.

FINS-Befehle ermöglichen die Kommunikation zwischen Teilnehmern in unterschiedlichen Netzwerken. Ein FINS-Master-Gerät kann die Trajexia-VR-Variablen und TABLE-Speichervariablen mit FINS-Befehlen lesen und schreiben. Diese Befehle benutzen den Ethernet-Anschluss des TJ1-MC__. Das FINS-Serverprotokoll ist ausschließlich UDP.

Hinweis
Die maximale Länge eines über eine Ethernet-Verbindung gegebenen FINS-Befehls beträgt 2012 Byte.

Trajexia benutzt folgende FINS-Befehle:
- 0101 (Speicher lesen)
- 0102 (Speicher schreiben)

Lesebefehl

Der FINS-Befehl **lesen** hat folgendes Format:

01	**01**	**..**	**..**	**..**	**00**	**..**	**..**
befehls_code		**var_typ**	**start_adresse**		**fest**	**element_zaehler**	

Die Parameter können die folgenden Werte haben:

Parameter	**Werte (hex)**
befehls_code	01 01
var_typ	• 82 (TABLE-Speicher im 16-Bit-Integerformat) • C2 (TABLE-Speicher im 32-Bit-IEEE-Gleitkommaformat) • B0 (VR-Speicher im 16-Bit-Integerformat)
start_adresse	0 <= **start_adresse** <= anzahl_variablen – 1 <= FFFF
Element_zähler	1 <= **element_zaehler** <= anzahl_variablen – **start_adresse**

Der TJ1-MC__ antwortet mit folgenden Codes:

Bedingung	Antwortcode (hex)	Beschreibung
Alle Elemente gültig	0000	OK
Var_typ ungültig	1101	Kein Bereichstyp
Start_adresse ungültig	1103	Adressenbereich-Spezifikationsfehler
Anzahl der Elemente ungültig	1104	Adresse außerhalb des Bereichs

Wenn **var_typ** 82 oder B0 und der Antwortcode 0000 ist, antwortet der TJ1-MC__ wie folgt:

01	01	00	00			
Befehls_code		**Antwort_code**		**wort_1**	**wort_2**	**...**

Wenn **var_typ** C2 und der Antwortcode 0000 ist, antwortet der TJ1-MC__ wie folgt:

01	01	00	00		
befehls_code		**antwort_code**		**dwort_1**	**...**

Hinweis
Die zurückgegebenen Worte und Datenworte sind im Big-Endian-Format.

Schreibbefehl

Der FINS-Befehl **schreiben** hat folgende Formate:

- Wenn **var_typ** 82 oder B0 ist:

01	02	..	..	..	00	..	..	..	..	..	..
befehls_code		**var_ typ**	**start_ adresse**		**fest**	**worte_gesamt**		**wort 1**		..	

- Wenn **var_typ** C2 ist:

01	02	C2	..	..	00	..	..	..	..	..	..	..
befehls_code		**var_typ**	**start_ adresse**		**fest**	**worte_gesamt**		**dwort 1**				..

- Wenn **var_typ** 30 ist:

01	02	30	..	..	00	..	..	..	..	
befehls_ code		**var_ typ**	**start_ adresse**		**bit_ num**	**bits_gesamt**		**bit**		

Die Parameter können die folgenden Werte haben:

Parameter	Werte
befehls_code	01 02
var_typ	• 82 (TABLE-Speicher im 16-Bit-Integerformat) • C2 (TABLE-Speicher im 32-Bit-IEEE-Gleitkommaformat) • B0 (VR-Speicher im 16-Bit-Integerformat) • 30 (VR-Speicher im 16-Bit-Format)
start_adresse	0 <= **start_adresse** <= Anzahl Variablen – 1 <= FFFF
worte_gesamt	1 <= **worte_gesamt** <= Speichergröße – **Start_adresse** + 1
bits_gesamt	1
bit	00 oder 01

Der TJ1-MC__ antwortet mit folgenden Codes:

Bedingung	Antwortcode (hex)	Beschreibung
Alle Elemente gültig	0000	OK
Var_typ ungültig	1101	Kein Bereichstyp
Start_adresse ungültig	1103	Adressenbereich-Spezifikationsfehler
Bit_nummer ungültig	1103	Adressenbereich-Spezifikationsfehler
Anzahl der Elemente ungültig (gesamt)	1104	Adresse außerhalb des Bereichs

4.2.5 FINS-Client-Protokoll

Trajexia kann die FINS-Kommunikation mit dem BASIC-Befehl **FINS_COMMS** initiieren. Einzelheiten dazu finden Sie in der Beschreibung der Befehle.

Sowohl der Befehle „Speicher lesen“ (0101) als auch der Befehl „Speicher schreiben“ (0102) werden unterstützt.

Diese Funktionalität ist nützlich für die Kommunikation mit einer OMRON-SPS, einem anderen Trajexia-System oder einem PC, auf dem die FINS-Server-Anwendungssoftware läuft.

Mit dem Befehl „Speicher lesen“ kann der Speicher aus anderen Geräten mit FINS-Serverfähigkeit gelesen werden. Der Befehl „Speicher schreiben“ kann zum Schreiben von Daten zu Geräten mit FINS-Serverfähigkeit verwendet werden.
Dieser Befehl gibt einen der folgenden Werte zurück, abhängig vom Ergebnis der Ausführung:

–1 Der Befehl wurde erfolgreich ausgeführt.

0 Der Befehl konnte nicht ausgeführt werden.

1 Die Abfrage wurde nicht gesendet, weil der Client oder das FINS-Protokoll belegt war.

2 Einer oder mehrere der Abfrageparameter ist/sind ungültig.

3 Ungültiger Quell-Speicherbereich.

4 Die Abfrage wurde gesendet, aber keine Antwort vom dezentralen Server innerhalb des Zeitüberschreitungs-Zeitraums empfangen.

5 Es wurde ein Fehlerantwort-Code vom dezentralen Server empfangen.

4.3 Serielles Protokoll

Der TJ1-MC__ TJ1-MC__ hat einen DB-9-Anschluss, der zwei serielle Schnittstellen enthält:
- Schnittstelle 1: RS232
- Schnittstelle 2: RS422 oder RS485, abhängig von den Schaltereinstellungen

Weitere Informationen finden Sie im Trajexia-Hardwarereferenzhandbuch

Beide Schnittstellen können unabhängig folgende Protokolle unterstützen:
- Host-Link-Master
- Host-Link-Slave
- Benutzerdefiniertes Protokoll

Hinweis
Die serielle Schnittstelle (Port 1) KANN NICHT zum Programmieren der Baugruppe verwendet werden.

4.3.1 Host-Link-Master

Wenn der TJ1-MC__ der Host-Link-Master ist, können Sie BASIC-Befehle an einen Host-Link-Slave senden, z. B. einen PC. Wenn Sie einen BASIC-Befehl an einen Host-Link-Slave senden, wird mit der Ausführung des nächsten BASIC-Befehls gewartet, bis der Host-Link-Slave eine Antwort sendet.

Sie können folgende BASIC-Befehle verwenden:

BASIC-Befehl	Beschreibung
HLM_COMMAND	**HLM_COMMAND** führt einen spezifischen Host Link-Befehl an den Slave aus.
HLM_READ	**HLM_READ** liest Daten aus dem Host-Link-Slave zum VR- oder TABLE-Speicher.
HLM_STATUS	**HLM_STATUS** gibt den Zustand des letzten Host Link-Masterbefehls an.
HLM_TIMEOUT	**HLM_TIMEOUT** definiert die Host-Link-Master-Zeitspanne für die Zeitüberschreitung.
HLM_WRITE	**HLM_WRITE** schreibt Daten aus dem VR- oder TABLE-Speicher zum Host-Link-Slave.
SETCOM	**SETCOM** konfiguriert die serielle Kommunikationsschnittstelle und aktiviert die Host-Link-Protokolle.

Befehle

Folgende Host-Link-Befehle werden für das Host-Link-Master-Protokoll unterstützt:

Typ	Header-Code	Name	Funktion
E/A-Speicher Lesen	**RR**	**CIO AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **CIO/IR**-Wort.
	RL	**LR AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **LR**-Wort.
	RH	**HR AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **HR**-Wort.
	RD	**DM AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **DM**-Wort.
	RJ	**AR AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **AR**-Wort.
	RE	**EM AREA READ**	Liest die spezifizierte Anzahl an Worten ab dem spezifizierten **EM**-Wort.
E/A-Speicher Schreiben	**WR**	**CIO AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **CIO/IR**-Wort.
	WL	**LR AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **LR**-Wort.
	WH	**HR AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **HR**-Wort.
	WD	**DM AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **DM**-Wort.
	WJ	**AR AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **AR**-Wort.
	WE	**EM AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten ab dem spezifizierten **EM**-Wort.

Typ	Header-Code	Name	Funktion
CPU-Baugruppe Status	**SC**	**STATUS WRITE**	Ändert die Betriebsart der CPU-Baugruppe.
Test	**TS**	**TEST**	Gibt einen vom Master gesendeten Datenblock unverändert zurück.
PC-Modell-Code lesen	**MM**	**PC MODEL READ**	Liest den Modell-Code der CPU-Baugruppe
Host-Link-Kommunikations-Verarbeitung	**XZ**	**ABORT** (nur Befehl)	Bricht die Operation ab, die durch einen Host-Link-Befehl ausgeführt wird und kehrt zum ursprünglichen Zustand zurück.
	**	**INITIALIZE** (nur Befehl)	Initialisiert die Übertragungssteue-rungsvorgänge für alle Host-Link-Baugruppen.
	IC	Undefinierter Befehl (nur Antwort)	Diese Antwort wird zurückgegeben, wenn der Header-Code eines Befehls ungültig ist.

Hinweis
Das Host-Link-Protokoll unterstützt nur C-Befehle. FINS wird nicht unterstützt.

Das Host-Link-Master-Protokoll unterstützt die Befehle nur in Einzelrahmen-Übertragung. Die folgende Tabelle zeigt, wie Sie das Host-Link-Protokoll mit den BASIC-Befehlen verwenden können und für welche Betriebsart der CPU-Baugruppe (RUN, MON oder PROG) der Befehl gültig ist.

Header-Code	Name	Benötigter BASIC-Befehl	RUN	MON	PRG
RR	**CIO AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
RL	**LR AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
RH	**HR AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
RD	**DM AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
RJ	**AR AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
RE	**EM AREA READ**	**HLM_READ**	Gültig	Gültig	Gültig
WR	**CIO AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
WL	**LR AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
WH	**HR AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
WD	**DM AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
WJ	**AR AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
WE	**EM AREA WRITE**	**HLM_WRITE**	Nicht gültig	Gültig	Gültig
SC	**STATUS CHANGE**	**HLM_COMMAND**	Gültig	Gültig	Gültig
TS	**TEST**	**HLM_COMMAND**	Gültig	Gültig	Gültig
MM	**PC MODEL READ**	**HLM_COMMAND**	Gültig	Gültig	Gültig
XZ	**ABORT** (nur Befehl)	**HLM_COMMAND**	Gültig	Gültig	Gültig
**	**INITIALIZE** (nur Befehl)	**HLM_COMMAND**	Gültig	Gültig	Gültig
IC	Undefinierter Befehl (Nur Antwort)	-	Gültig	Gültig	Gültig

Achtung
Die Host-Link-Masterbefehle dürfen nur von einem Programm ausgeführt werden, um Timing-Probleme mit Befehlen in Programmen, die in anderen Tasks laufen, zu vermeiden.

Achtung
Die Host-Link-Masterbefehle stellen die Werkzeuge zum Austausch von Daten mit dem Host-Link-Slave bereit. Das Anwenderprogramm muss ordnungsgemäße Fehlerbehandlungsroutinen enthalten, um Kommunikationsfehler zu beheben und nötigenfalls Wiederholungsversuche durchzuführen.

Endcodes

Folgende Endcodes sind im Parameter **HLM_STATUS** definiert:

Endcode	Beschreibung	Mögliche Ursache	Lösung
$00	Normale Ausführung	Es liegt kein Problem vor.	–
$01	Nicht ausführbar in der RUN-Betriebsart	Der gesendete Befehl kann nicht ausgeführt werden, wenn die SPS in der RUN-Betriebsart ist.	Überprüfen Sie die Beziehung zwischen dem Befehl und der SPS-Betriebsart.
$13	FCS-Fehler	FCS ist falsch.	Beeinträchtigung durch Störungen; übertragen Sie den Befehl noch einmal.
$14	Formatfehler	• Das Befehlsformat ist falsch. • Ein unteilbarer Befehl ist geteilt worden. • Die Rahmenlänge ist kleiner als die Mindestlänge für den betreffenden Befehl.	Überprüfen Sie das Format, und übertragen Sie den Befehl noch einmal.
$15	Dateneingabe-Fehler	Die Daten liegen außerhalb des spezifizierten Bereichs oder sind zu lang.	Korrigieren Sie die Befehlsargumente, und übertragen Sie den Befehl noch einmal.
$18	Rahmenlänge-Fehler	Die maximal zulässige Rahmenlänge von 131 Byte wurde überschritten.	Überprüfen Sie den Befehl, und übertragen Sie ihn noch einmal.
$19	Nicht ausführbar	Sie haben keine Zugriffsberechtigung.	Erlangen Sie eine Zugriffsberechtigung.
$21	Nicht ausführbar wegen CPU-Fehler.	Der Befehl kann nicht ausgeführt werden, weil ein CPU-Fehler aufgetreten ist.	Schalten Sie die Spannungsversorgung der CPU aus und anschließend wieder ein.
$100	Host-Link-Slave ACK-Zeitüberschreitung	–	–
$200	Adressenfehler undefinierter Befehl	–	–

Einstellen

Sie brauchen den Befehl **SETCOM**, um die serielle Schnittstelle für das Host-Link-Masterprotokoll des TJ1-MC__ einzustellen. Stellen Sie den Befehl wie folgt ein:

SETCOM(baud_rate, daten_bits, stopp_bits, paritaet, schnittstelle, 6)

Nach dieser Einstellung können Sie die Befehle **HLM_READ**, **HLM_WRITE** und **HLM_COMMAND** zum Lesen und Schreiben von Daten über Host-Link verwenden.

Zeitüberschreitung

Der Zeitüberschreitungsmechanismus ist implementiert, um zu verhindern, dass der BASIC-Task auf Grund von schlechter oder fehlender Kommunikation zu lange angehalten wird. Der Parameter **HLM_TIMEOUT** spezifiziert den Zeitraum für die Zeitüberschreitung. Dieser Zeitraum ist die maximale Zeit, die der Programm-Task auf eine Antwort wartet, nachdem er den Befehl gesendet hat.
Wenn die Zeit für die Zeitüberschreitung abgelaufen ist, enthält **HLM_STATUS** den Status des Befehls, und der BASIC-Task wird fortgesetzt.
Der Parameter **HLM_TIMEOUT** spezifiziert die für die Zeitüberschreitung festgelegte Zeit für alle Befehle und alle Schnittstellen.

Status

Der Parameter **HLM_STATUS** enthält den Status des letzten Host-Link-Masterbefehls, der an die spezifizierte Schnittstelle gesendet wurde.
Der Parameter zeigt den Status für die Befehle **HLM_READ**, **HLM_WRITE** und **HLM_COMMAND** an. Die Status-Bits sind:

Bit	Name	Beschreibung
0–7	Endcode	Der Endcode ist: • der vom Host-Link-Slave definierte Endcode, wenn ein Problem in der Daten-Zeichenkette oder dem gesendeten Befehl aufgetreten ist oder • ein vom Host-Link-Master definierter Endcode, wenn ein Problem in der Daten-Zeichenkette der empfangenen Antwort aufgetreten ist.
8	Zeitüberschreitungs-fehler	Ein Zeitüberschreitungsfehler tritt auf, wenn innerhalb der für die Zeitüberschreitung festgelegten Zeit keine Antwort empfangen wird. Dies deutet daraufhin, dass die Kommunikation unterbrochen ist.
9	Befehl nicht erkannt	Dieser Zustand deutet darauf hin, dass der Slave den Befehl nicht erkannt und eine IC-Antwort zurückgegeben hat.

Der Parameter HLM_STATUS hat den Wert 0, wenn keine Probleme aufgetreten sind. Ist der Wert nicht 0, muss eine geeignete Maßnahme, z. B. ein Neuversuch oder Nothalt im Anwender-BASIC-Programm programmiert werden. Jede Schnittstelle hat einen HLM_STATUS-Parameter. Der Modifikator **PORT** ist erforderlich, um die Schnittstelle zu spezifizieren.

Beispiele

In diesen Beispielen gehen wir von folgender Einrichtung aus:
- Ein Trajexia-System mit einem TJ1-MC__.
- Ein Slave-PC mit Teilnehmeradresse 13.
- Eine Verbindung von der seriellen Schnittstelle des TJ1-MC__ zum PC. Die serielle Schnittstelle verwendet RS422-Kommunikation.

Beispiel	Lesen von Daten aus dem PC mit **HLM_READ**.
BASIC-Code	**' Host-Link-Master für Schnittstelle 2 einstellen** **SETCOM(9600,7,2,2,2,6)** **' Quelladresse: CIO/IR 002** **' Datenmenge: 2 Worte** **' Zieladresse: VR(0)** **HLM_READ(2,13,PLC_IR,2,2,MC_VR,0)**
Host-Link-Kommunikation	• Vom Host-Link-Master zum Host-Link-Slave: @13RR0002000242* • Vom Host-Link-Slave zum Host-Link-Master: @13RR000101010241*
Ergebnis	• VR Adresse = 0: Wert = 257.0000 • VR Adresse = 1: Wert = 258.0000
Beispiel	Schreiben von Daten zum PC mit **HLM_WRITE**.
BASIC-Code	**' Quelladresse: TABLE(18)** **' Datenmenge: 2 Worte** **' Zieladresse: LR 014** **TABLE(18,$0701,$0702)** **HLM_WRITE(2,13,PLC_LR,14,2,MC_TABLE,18)**
Host-Link-Kommunikation	• Vom Host-Link-Master zum Host-Link-Slave: @13WL0014070107025F* • Vom Host-Link-Slave zum Host-Link-Master: @13WL0059*
Ergebnis	• LR-Adresse = 0: Wert = 701 (hex) • LR-Adresse = 1: Wert = 702 (hex)

Beispiel	Senden eines **TS**- (Test-) Befehls zum PC mit **HLM_COMMAND**.
BASIC-Code	**HLM_COMMAND(HLM_TEST,2,13)**
Host-Link-Kommunikation	• Vom Host-Link-Master zum Host-Link-Slave: @13TSMCW151 TEST STRING2A* • Vom Host-Link-Slave zum Host-Link-Master: @13TSMCW151 TEST STRING2A*
Ergebnis	**HLM_STATUS PORT(2) = 0**, was auf eine korrekte Kommunikation hinweist.

Beispiel	Den PC mit **HLM_COMMAND** in die MON-Betriebsart setzen.
BASIC-Code	**HLM_COMMAND(HLM_STWR,2,13,2)**
Host-Link-Kommunikation	• Vom Host-Link-Master zum Host-Link-Slave: @13SC0250* • Vom Host-Link-Slave zum Host-Link-Master: @13SC0052*
Ergebnis	Der PC läuft in der MON-Betriebsart. Bitte beachten Sie, dass Sie **HLM_WRITE** benutzen müssen, wenn Sie Daten zum PC schreiben wollen.

Beispiel	Den PC-Modellcode mit **HLM_COMMAND** (Zeitüberschreitung) auslesen.
BASIC-Code	**HLM_TIMEOUT=500** **' Zieladresse: VR(100)** **HLM_COMMAND(HLM_MREAD,2,13,MC_VR,100)**
Host-Link-Kommunikation	• Vom Host-Link-Master zum Host-Link-Slave: @13MM42* • Vom Host-Link-Slave zum Host-Link-Master: keine Antwort
Ergebnis	Weil der Master keine Antwort vom PC erhielt, hat **HLM_STATUS PORT(2)** nach 500 Servozyklen den Wert 256 (Bit 8 ist gesetzt).

4.3.2 Host-Link-Slave

Wenn der TJ1-MC__ der Host-Link-Slave ist, kann ein Host-Link-Master (z. B. ein programmierbares Bedienterminal) Daten aus dem TJ1-MC__ lesen und Daten zu ihm schreiben. Das Mapping zwischen dem Slave und dem Master ist:

TJ1-MC__-Speicher	Host-Link-Mapping	Adressbereich
VR	**CIO**	0 bis 1023
TABLE	**DM**	0 bis 63999

Sie können folgende BASIC-Befehle verwenden:

BASIC-Befehl	Beschreibung
SETCOM	**SETCOM** konfiguriert die serielle Kommunikationsschnittstelle und aktiviert die Host-Link-Protokolle.
HLS_NODE	Der Parameter **HLS_NODE** definiert die Slave-Baugruppennummer für das Host-Link-Slave-Protokoll.
HLS_MODEL	**HLS_MODEL** definiert den Modellcode des TJ1-MC__ für das Host-Link-Slave-Protokoll.

Befehle

Die für das Host-Link-Slaveprotokoll unterstützten Befehle sind in der Tabelle unten angegeben. Das Protokoll unterstützt Einzelrahmen- und Mehrfachrahmen-Übertragung.

Typ	Header-Code	Name	Funktion
E/A-Speicher lesen	**RR**	**CIO AREA READ**	Liest die spezifizierte Anzahl an Worten aus dem VR-Speicher, beginnend mit dem spezifizierten Wort.
	RD	**DM AREA READ**	Liest die spezifizierte Anzahl an Worten aus dem TABLE-Speicher, beginnend mit dem spezifizierten Wort.

Typ	Header-Code	Name	Funktion
E/A-Speicher schreiben	**WR**	**CIO AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten in den VR-Speicher, beginnend mit dem spezifizierten Wort.
	WD	**DM AREA WRITE**	Schreibt die spezifizierten Daten in Worteinheiten in den TABLE-Speicher, beginnend mit dem spezifizierten Wort.
Test	**TS**	**TEST**	Gibt einen vom Master gesendeten Datenblock unverändert zurück.
PC-Modellcode lesen	**MM**	**PC MODEL READ**	Liest den Modellcode des TJ1-MC__, wie er durch den Parameter **HLS_MODEL** spezifiziert ist.
E/A-Speicherbereichregistrierung und Lesen	**QQMR**	**REGISTER I/O MEMORY**	Registriert die E/A-Tabelle mit dem Inhalt der aktuellen E/A-Konfiguration.
	QQIR	**READ I/O MEMORY**	Liest alle registrierten Worte und Bits gleichzeitig aus dem E/A-Speicher.
Host-Link-Kommunikationsverarbeitung	**XZ**	**ABORT** (nur Befehl)	Bricht die Operation ab, die durch einen Host-Link-Befehl ausgeführt wird und kehrt zum ursprünglichen Zustand zurück.
	**	**INITIALIZE** (nur Befehl)	Initialisiert die Übertragungs-steuerungsvorgänge für alle Host-Link-Baugruppen.
	IC	Undefinierter Befehl (nur Antwort)	Diese Antwort wird zurückgegeben, wenn der Header-Code eines Befehls ungültig ist.

Endcodes

Dies sind die Antwort-Endcodes, die im Antwort-Rahmen zurückgegeben werden:

Endcode	Beschreibung	Mögliche Ursache	Lösung
0	Normale Ausführung	Es liegt kein Problem vor.	–
13	FCS-Fehler	FCS ist falsch.	Überprüfen Sie die FCS-Berechnungsmethode. Wenn es Beeinträchtigung durch Störungen gab, übertragen Sie den Befehl noch einmal.
14	Formatfehler	• Das Befehlsformat ist falsch. • Ein unteilbarer Befehl ist geteilt worden. • Die Rahmenlänge ist kleiner als die Mindestlänge für den betreffenden Befehl.	Überprüfen Sie das Format, und übertragen Sie den Befehl noch einmal.
15	Dateneingabe-Fehler	Die Daten liegen außerhalb des spezifizierten Bereichs oder sind zu lang.	Korrigieren Sie die Befehlsargumente, und übertragen Sie den Befehl noch einmal.
18	Rahmenlänge-Fehler	Die maximal zulässige Rahmenlänge von 131 Byte wurde überschritten.	Überprüfen Sie die Daten, und übertragen Sie den Befehl noch einmal.
19	Nicht ausführbar	Ein E/A-Speicher-Batch wurde ausgeführt, als zu lesende Elemente nicht registriert waren.	Registrieren Sie die Elemente, die gelesen werden sollen, bevor Sie einen Batch-Leseversuch unternehmen.
A3	Abbruch auf Grund eines FCS-Fehlers in den Übertra-gungsdaten	Im zweiten oder einem späteren Rahmen trat ein FCS-Fehler auf.	Korrigieren Sie die Befehlsdaten, und übertragen Sie den Befehl noch einmal.

Endcode	Beschreibung	Mögliche Ursache	Lösung
A4	Abbruch auf Grund eines Formatsfehlers in den Übertragungsdaten	Das Befehlsformat stimmte nicht mit der Byte-Anzahl im zweiten oder einem späteren Rahmen überein.	Korrigieren Sie die Befehlsdaten, und übertragen Sie den Befehl noch einmal.
A5	Abbruch auf Grund eines Dateneingabefehlers in den Übertragungsdaten	Es gab einen Dateneingabefehler oder einen Fehler in der Datenlänge im zweiten oder einem späteren Rahmen.	Korrigieren Sie die Befehlsdaten, und übertragen Sie den Befehl noch einmal.
A8	Abbruch auf Grund eines Rahmenlängefehlers in den Übertragungsdaten	Die Länge des zweiten oder einen späteren Rahmens hat die maximal zulässige Länge von 128 Byte überschritten.	Korrigieren Sie die Befehlsdaten, und übertragen Sie den Befehl noch einmal.

Einstellen

Sie brauchen den Befehl **SETCOM**, um die serielle Schnittstelle für das Host-Link-Slaveprotokoll des TJ1-MC__ einzustellen. Stellen Sie den Befehl wie folgt ein:

SETCOM(baud_rate, daten_bits, stopp_bits, paritaet, schnittstelle, 5)

Nachdem dieser Befehl gegeben wurde, antwortet der TJ1-MC__ auf die Host-Link-Befehle vom Master mit der spezifizierten Teilnehmernummer. Sie können diese Teilnehmernummermit dem Parameter **HLS_NODE** einstellen.

Beispiel

In diesen Beispielen gehen wir von folgender Einrichtung aus:

- Ein Trajexia-System mit einem TJ1-MC__.
- Ein programmierbares Bedienterminal Typ NS8.
- Eine Verbindung von der seriellen Schnittstelle des TJ1-MC__ zum programmierbaren Bedienterminal. Die serielle Schnittstelle verwendet RS232C-Kommunikation.

Beispiel	Konfiguration des Host-Link-Slave.
BASIC-Code	**' Definiert Host-Link-Slave-Teilnehmer** **HLS_NODE = 15** **' Definiert Host-Link-Slave-Modellcode** **HLS_MODEL = $FA** **' Host-Link-Slave für Schnittstelle 1 einstellen** **SETCOM(9600,7,2,2,1,5)**
Ergebnis	Der TJ1-MC__ kann mit dem programmierbarem Bedienterminal kommunizieren.

4.3.3 Benutzerdefiniertes Protokoll

Mit folgenden Befehlen können es Sie ein benutzerdefiniertes Kommunikationsprotokoll implementieren:

BASIC-Befehl	Beschreibung
SETCOM	**SETCOM** konfiguriert die serielle Kommunikationsschnittstelle und aktiviert die Host-Link-Protokolle.
GET	Der Befehl **GET** ordnet den ASCII-Code eines empfangenen Zeichens einer Variablen zu.
INPUT	Der Befehl **INPUT** ordnet den spezifizierten Variablen numerische Eingabe-Zeichenkettenwerte zu.
KEY	**KEY** gibt **TRUE** oder **FALSE** zurück, wenn ein Zeichen empfangen bzw. nicht empfangen wurde.
LINPUT	Der Befehl **LINPUT** ordnet den ASCII-Code eines empfangenen Zeichens einer Anordnung von Variablen zu.
PRINT	**PRINT** gibt eine Zeichenserie an ein serielles Ausgabegerät aus.

Beispiel

Folgende Einrichtung wird angenommen:

- Ein Trajexia-System mit einem TJ1-MC__.
- Ein OMRON F500-Bildverarbeitungssystem.
- Eine Verbindung von der seriellen Schnittstelle des TJ1-MC__ zum F500. Die serielle Schnittstelle verwendet RS232C-Kommunikation (Schnittstelle 1).

Dieses Programm sendet über die serielle Schnittstelle einen Bildverarbeitungsbefehl, liest die Antwort vom Bildverarbeitungssystem, schreibt sie zu VR-Variablen und gibt das Ergebnis im Terminalfenster von Trajexia Tools aus.

```
' Im STARTUP-Programm
' Einstellung der RS232-Schnittstelle für das Bildverarbeitungssystem
SETCOM(38400,8,1,0,1,0)
' Im Anwendungsprogramm
schleife:
   ' Trigger, steigende Flanke im virtuellen System
   WAIT UNTIL IN(30)=0
   WAIT UNTIL IN(30)=1
   ' Bildschirm loeschen
   PRINT CHR(27);"[2J"

   ' Befehl loeschen
   GOSUB befehl_loeschen

   ' Befehl gemaess VR(10) an die serielle Schnittstelle senden
   IF bildverarbeitungs_befehl=v_messen THEN
      PRINT #1, "M"
      PRINT ">> M"
   ELSEIF bildverarbeitungs_befehl=v_datum THEN
      PRINT #1, "DATE"
      PRINT ">> DATE"
   ELSEIF bildverarbeitungs_befehl=v_szene THEN
      PRINT #1,"SCENE ";szene_n
      PRINT ">> SCENE"
   ENDIF

   'Antwort prüfen
   GOSUB befehl_lesen

GOTO schleife
   befehl_lesen:
   zaehler=0
   antw_status=0
   k=-1
   TICKS=5000
   REPEAT
      IF KEY#1 THEN
         zaehler= zaehler+1
         GET#1, k
         'PRINT k;zaehler
         TABLE(zaehler,k)
         'PRINT zaehler
      ENDIF
   UNTIL TICKS<0 'OR k=13
   PRINT "Empfangen ";zaehler[0];" Zeichen"
   FOR i=1 TO zaehler
      IF TABLE(i)<>13 THEN
         PRINT CHR(TABLE(i))
      ELSE
         PRINT "'Zn'"
      ENDIF
   NEXT i
   IF TICKS<0 THEN
      PRINT "Zeitueberschreitung bei Kommunikation mit F500"
      antw_status=3
   ELSEIF TABLE(zaehler-2)=79 AND TABLE(zaehler-1)=75 THEN
      PRINT "Antwort OK"
      antw_status=1
   ELSE
      PRINT "Antwort falsch"
      antw_status=2
   ENDIF
   PRINT "Antwortstatus ist :";antw_status[0]
RETURN
befehl_loeschen:
   PRINT "Wird geloescht..."
   WHILE KEY#1
      GET#1,k
      PRINT k
   WEND
   PRINT "Geloescht!!"
RETURN
```

4.4 PROFIBUS

4.4.1 Einführung

PROFIBUS ist ein internationaler offener Feldbusstandard.
Der TJ1-PRT ermöglicht die Kommunikation zwischen dem Trajexia-System und einem PROFIBUS-Netzwerk. Er tauscht Daten zwischen dem PROFIBUS-Master und dem TJ1-MC__ aus.
Zu diesem Zweck benutzt er Trajexia VR-Variablen.

4.4.2 Einrichtung der Kommunikation

Der TJ1-PRT hat zwei Teilnehmernummer-Wählschalter.
Sie können die Teilnehmernummer-Wählschalter verwenden, um dem TJ1-PRT eine PROFIBUS-Netzwerkadresse zuzuweisen.
Sie müssen dem TJ1-PRT eine Adresse zuweisen, bevor Sie die Spannungsversorgung des Trajexia-Systems einschalten.

Um den TJ1-PRT zu initialisieren, verwenden Sie den BASIC-Befehl **PROFIBUS**:

PROFIBUS(baugruppen_nummer, 2, 1, ausgangs_start, ausgangs_zaehler, eingangs_start, eingangs_zaehler)

Erläuterungen:
- **baugruppen_nummer** ist die Nummer der TJ1-PRT-Baugruppe.
- **ausgangs_start** ist die Startadresse des Ausgangsdatenbereichs von VR-Variablen.
- **ausgangs_zaehler** ist die Anzahl von VR-Variablen im Ausgangsdatenbereich.
- **eingangs_start** ist die Startadresse des Eingangsdatenbereichs von VR-Variablen.
- **eingangs_zaehler** ist die Anzahl von VR-Variablen im Eingangsdatenbereich.

Hinweis
Die maximale Anzahl von VR-Variablen zum Datenaustausch ist 122.

Nachdem der Befehl **PROFIBUS(baugruppen_nummer, 2, ...)** ausgeführt wurde, werden die Daten-Arrays automatisch ausgetauscht. Die zwischen dem TJ1-PRT und dem PROFIBUS-Master ausgetauschten Daten sind im 16-Bit-Integerformat. Jedes ausgetauschte Wort reicht von –32768 bis 32767.

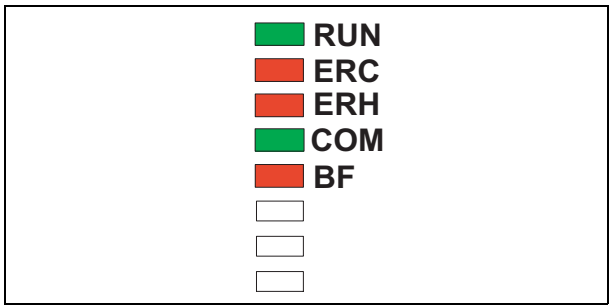
Eine VR-Variable kann eine 24-Bit-Nummer speichern; sie kann ebenfalls Fragmente speichern. Der Austausch mit dem PROFIBUS-Master unterstützt keine Werte außerhalb des Bereichs von –32768 bis 32767 sowie Fragmente.

Eine Beispielsequenz zum Konfigurieren der TJ1-PRT-Baugruppe sieht wie folgt aus:

1. Stellen Sie die Einheitenummer mit den beiden Drehschaltern auf der Vorderseite der TJ1-PRT-Baugruppe ein.
2. Schalten Sie die Spannungsversorgung für das System ein. Die **RUN**-LED leuchtet auf. Die **ERH**-LED blinkt.
3. Erstellen Sie ein BASIC-Programm, das den Befehl **PROFIBUS(2,2,1,10,7,150,3)** enthält. In diesem Beispiel initialisiert das System eine TJ1-PRT-Baugruppe mit der Baugruppennummer 2. Das System sendet 7 Ausgangsworte vom Master an die VR's 10 bis 16 und drei Eingangsworte von den VRs 150 bis 152 an den Master.
4. Wenn die Konfiguration erfolgreich ist, leuchten die **RUN**-LED und die **COMM**-LED auf. Die Kommunikation ist jetzt aktiv.

Um den CJ1-PRM21 mit dem CX-PROFIBUS zu konfigurieren, führen Sie folgende Schritte durch:

Abb.5

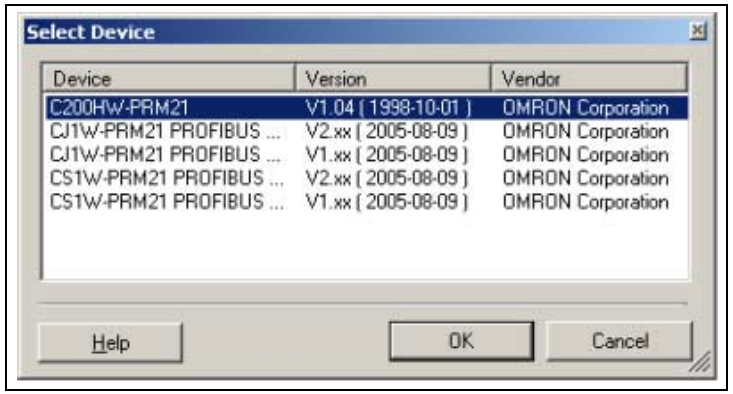


1. Starten Sie das CX-PROFIBUS-Software-Tool.
2. Klicken Sie mit der rechten Maustaste auf den Baum „Eigenes Netzwerk“.
3. Wählen Sie **Gerät hinzufügen...**.

Abb.6

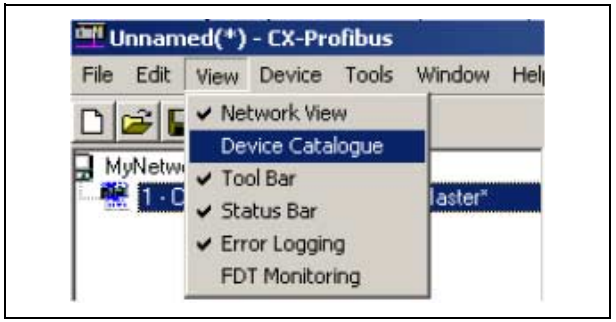
4. Wählen Sie das PROFIBUS-Masterboard.
5. Klicken Sie auf **OK**.

Abb.7



6. Öffnen Sie den **Gerätekatalog** aus dem Menü **Ansicht**.

Abb.8



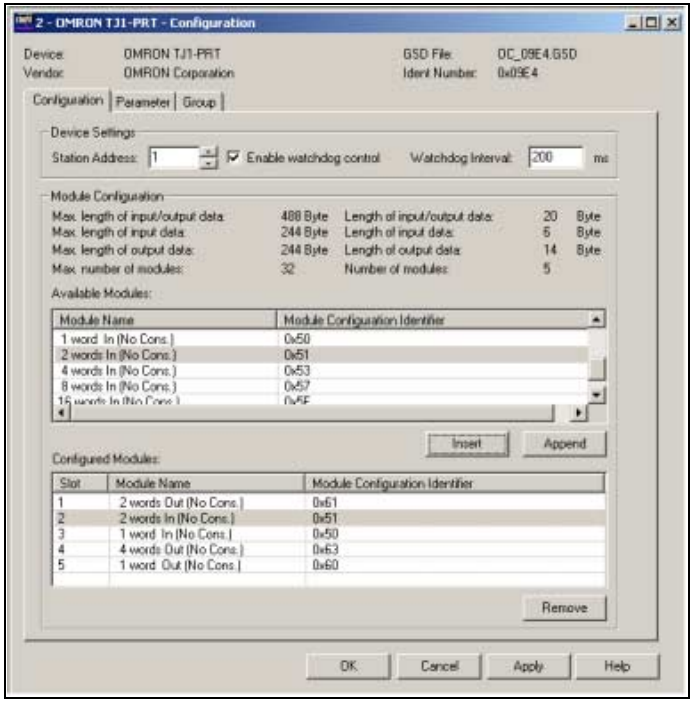
7. Klicken Sie auf **GSD-Dateien installieren...**. Die GSD-Datei befindet sich auf der Trajexia Tools-CD. Sie können Sie auch aus dem Download Center auf der OMRON-Website herunterladen.
8. Klicken Sie auf **Update**. Der TJ1-PRT wird in der Liste angezeigt.
9. Wählen Sie den OMRON TJ1-PRT aus der Liste, und klicken Sie auf **Gerät hinzufügen**.

Abb.9

10. Doppelklicken Sie auf das TJ1-PRT-Slavemodul im Baum „Eigenes Netzwerk“.
11. Geben Sie die Teilnehmernummer in das Feld **Stationsadresse** ein.
12. Fügen Sie mit **Einfügen** Eingangs- und Ausgangsmodule in die Konfigurationsliste unten ein.
13. Stellen Sie sicher, dass die Menge der Eingangs- und Ausgangsworte in den ausgewählten Modulen gleich der Menge ist, die mit dem PROFIBUS-Befehl ausgewählt wurde.
14. Klicken Sie auf **OK**.

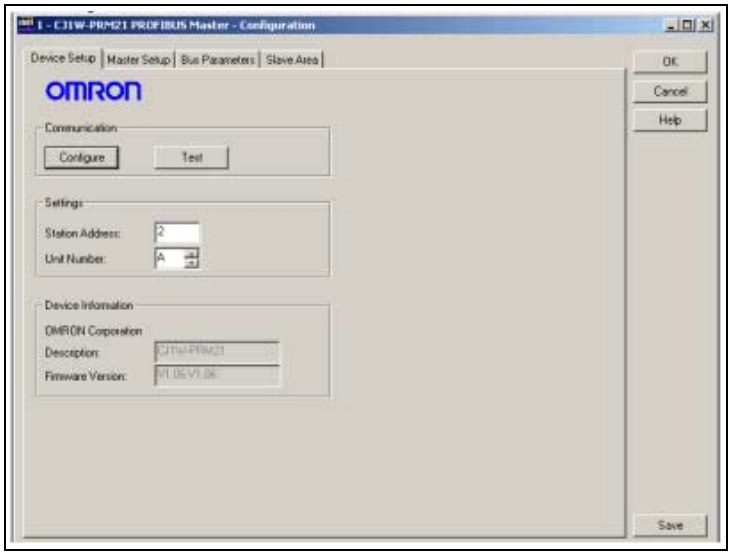
Um den CJ1W-PRM21 mit dem CX-PROFIBUS zu konfigurieren, führen Sie folgende Schritte durch:

Abb.10



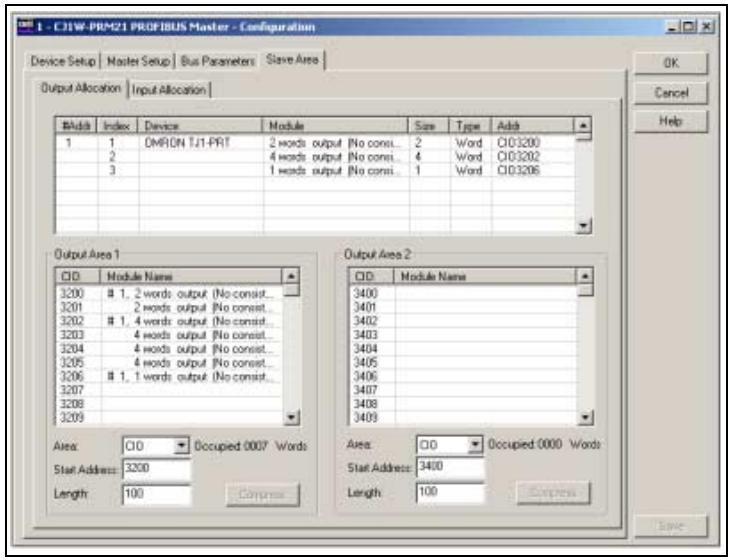
1. Doppelklicken Sie auf das Mastermodul im Baum „Eigenes Netzwerk“.
2. Geben Sie die **Stationsadresse** und **Baugruppennummer** ein.

Abb.11



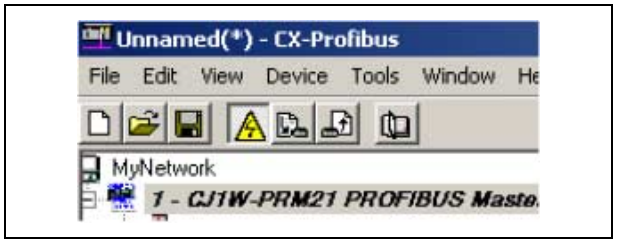
3. Klicken Sie auf das Register **Slave-Bereich**.
4. Geben Sie die **Startadresse** im entsprechenden Feld von **Ausgangsbereich 1** und **Eingangsbereich 1** ein.
5. Speichern Sie das Projekt.

Abb.12



6. Klicken Sie auf die Schaltfläche **Gerät online/offline (Umschalter)**, um online zu gehen.
7. Klicken Sie auf die Schaltfläche **Geräte-Download**, um die Parameter herunterzuladen.

Abb.13



4.4.3 Kommunikationsstatus

Der TJ1-PRT kann Statusinformationen für den TJ1-MC__ liefern. Sie können die Statusinformationen in BASIC mit dem Befehl **PROFIBUS** (baugruppen_nummer,4,0) abrufen. Als Ergebnis werden folgende Informationen angezeigt:

Bit	Wert	Beschreibung
0	0	Konfiguration des E/A-Datenaustausches fehlgeschlagen
	1	E/A-Datenaustausch erfolgreich konfiguriert
1	0	E/A-Daten nicht verfügbar
	1	E/A-Daten verfügbar
2	0	Datenaustausch in der Betriebsart OPERATE ist aktiv
	1	Datenaustausch in der Betriebsart CLEAR ist aktiv

4.5 DeviceNet

4.5.1 Einführung

DeviceNet ist ein internationaler offener Feldbusstandard, der auf dem CAN-Protokoll basiert. Der TJ1-DRT ermöglicht die Kommunikation zwischen dem Trajexia-System und einem DeviceNet -Netzwerk.
Er tauscht Daten zwischen einem DeviceNet -Master und dem TJ1-MC__ aus. Zu diesem Zweck benutzt er Trajexia VR-Variablen.

4.5.2 Einrichtung der Kommunikation

Der TJ1-DRT hat zwei Teilnehmernummer-Wählschalter. Sie können die Auswahlfunktion für die Teilnehmernummer verwenden, um dem TJ1-DRT eine Teilnehmernummer zuzuweisen.

Die DeviceNet-Teilnehmernummern haben einen Bereich von 0 bis 63. Wenn Sie mit den Wählschaltern eine Teilnehmernummer wählen, die diesen Bereich überschreitet, wird die von der Software vorgegebene Teilnehmernummer verwendet. Die Teilnehmer, welche Software-Einstellungen erlauben, sind 64 bis 99.

Um den TJ1-DRT zu initialisieren, verwenden
Sie den BASIC-Befehl **DEVICENET**:

DEVICENET(baugruppen_nummer, 2, 1, ausgangs_start, ausgangs_zaehler, eingangs_start, eingangs_zaehler)

Erläuterungen:
- **baugruppen_nummer** ist die Nummer der TJ1-DRT-Baugruppe.
- **ausgangs_start** ist die Startadresse des Ausgangsdatenbereichs von VR-Variablen.
- **ausgangs_zaehler** ist die Anzahl von VR-Variablen im Ausgangsdatenbereich.
- **eingangs_start** ist die Startadresse des Eingangsdatenbereichs von VR-Variablen.
- **eingangs_zaehler** ist die Anzahl von VR-Variablen im Eingangsdatenbereich.

Hinweis
Die maximale Anzahl von VR-Variablen zum Datenaustausch ist 32.

Hinweis
Wenn Sie einen OMRON DeviceNet-Master verwenden, ist es ratsam, für den eingangs_zaehler oder den ausgangs_zaehler einen Wert von 4,8,16 oder 32 für die VR-Variablen zu wählen.

Nachdem der Befehl **DEVICENET(baugruppen_nummer, 2, ...)** ausgeführt wurde, werden die Daten-Arrays automatisch ausgetauscht. Die zwischen dem TJ1-DRT und dem DeviceNet -Master ausgetauschten Daten sind im 16-Bit-Integerformat. Jedes ausgetauschte Wort reicht von –32768 bis 32767.

Eine VR-Variable kann eine 24-Bit-Nummer speichern; sie kann ebenfalls Fragmente speichern. Der Austausch mit dem DeviceNet -Master unterstützt keine Werte außerhalb des Bereichs von –32768 bis 32767 oder Fragmente.

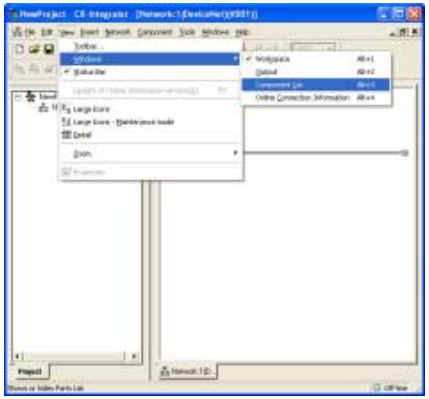
DeviceNet-Netzwerk konfigurieren
Um den OMRON CJ1W/CS1W-DRM21 DeviceNet-Master für den Austausch von VR-Variablen mit dem Trajexia-System zu konfigurieren, führen Sie folgende Schritte durch:

1. Starten Sie CX-Integrater im CX-ONE-Software-Tool.
2. Wählen Sie „Netzwerk“ aus dem Menü „Einfügen“.
3. Wählen Sie „DeviceNet“ aus dem Bildschirm „Netzwerk auswählen“. Die Netzwerkansicht wird angezeigt.

Abb.14

4. Wählen Sie „CJ1W-DRM21“ aus der OMRON-Kommunikationsadapter-Liste.

Abb.15



5. Ziehen Sie den CJ1W-DRM21 mit der Maus in das Netzwerkfenster, und legen Sie ihn dort ab.

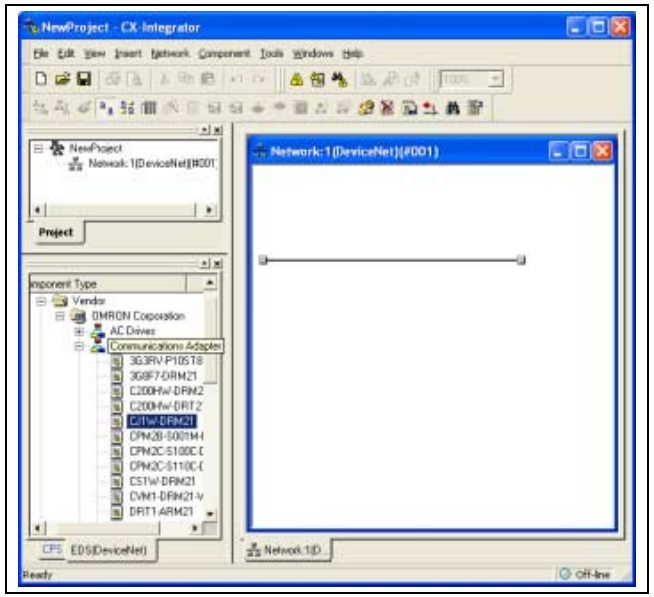
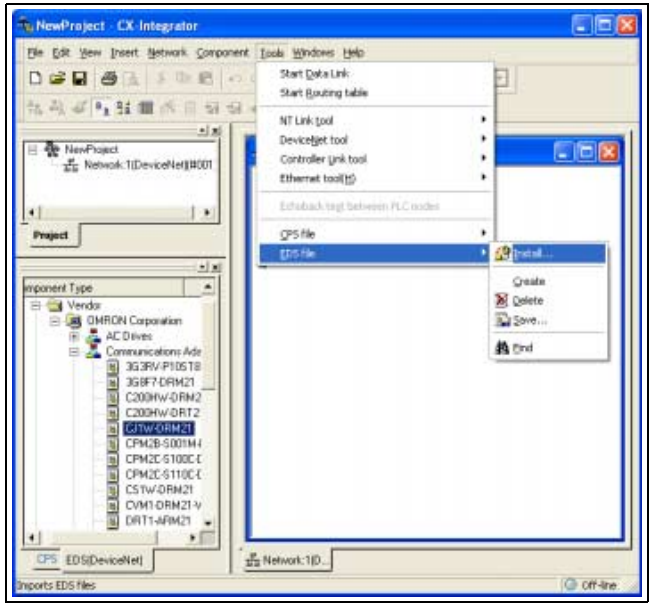


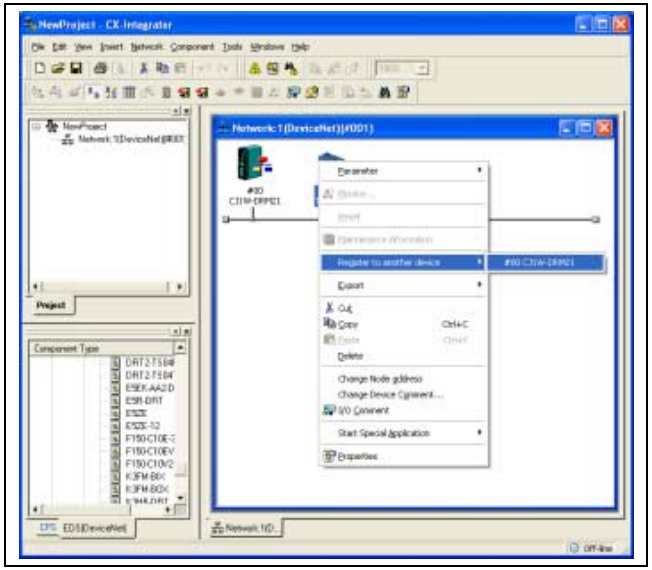
Abb.16

6. Installieren Sie die **EDS**-Datei aus dem CX-Integrator.
7. Wählen Sie im Dialogfeld „Nein“. Das Symbol wird nicht benötigt.



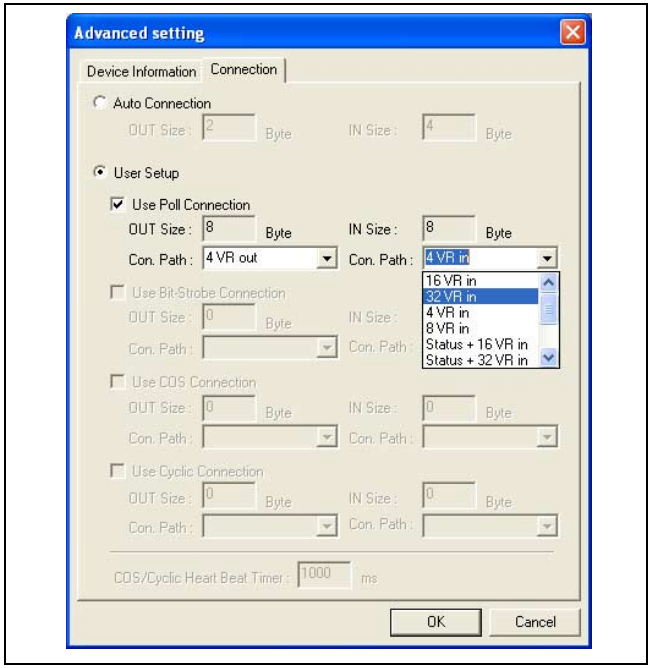
8. Registrieren Sie den Slave für den Master; klicken Sie mit der rechten Maustaste auf das #01TJ1-DRT-Symbol.
9. Doppelklicken Sie auf das Master-Symbol.
10. Wählen Sie als Gerät „TJ1-DRT“ aus.

Abb.17



11. Klicken Sie auf „Erweitertes Setup“.
12. Klicken Sie auf das Register „Verbindung“.
13. Klicken Sie auf „Benutzer-Setup“.
14. Klicken Sie auf „Poll-Verbindung verwenden“.
15. Wählen Sie „Verb.-pfad“.
16. Wählen Sie die Anzahl der Variablen, die für die DeviceNet-Kommunikation gewählt wurde.
17. Klicken Sie auf OK, um alle Dialogfelder zu bestätigen.
18. Wählen Sie aus dem Netzwerk-Menü „Online arbeiten“.
19. Wählen Sie Parameter aus dem Komponentenmenü aus.
20. Klicken Sie mit der rechten Maustaste auf das Master-Symbol.
21. Wählen Sie „Parameter-Download“.

Abb.18



4.5.3 Kommunikationsstatus

Der TJ1-DRT kann Statusinformationen sowohl für den TJ1-MC__ als auch für den DeviceNet-Master bereitstellen. Sie können die Statusinformationen in BASIC mit dem Befehl DeviceNet (baugruppen_nummer,4,0) abrufen. Als Ergebnis werden folgende Informationen angezeigt:

Bit	Wert	Beschreibung
0	0	DeviceNet (baugruppen_nummer, 2, ...) noch nicht ausgeführt
	1	DeviceNet (baugruppen_nummer, 2, ...) mit Fehler ausgeführt
1	0	Keine DeviceNet-E/A-Verbindung
	1	DeviceNet-E/A-Verbindung läuft
2	0	VR-Variablen im Ausgangsdatenbereich sind aktualisiert worden
	1	VR-Variablen im Ausgangsdatenbereich sind noch nicht aktualisiert worden
3	0	Die DeviceNet-E/A-Verbindungsgröße stimmt mit dem Befehl DeviceNet (baugruppen_nummer, 2, …) überein.
	1	Die DeviceNet-E/A-Verbindungsgröße stimmt nicht mit dem Befehl DeviceNet (baugruppen_nummer, 2, …) überein.
4–7	0	Immer null
8	0	Netzwerk-Spannungsversorgung OK
	1	Netzwerk-Spannungsversorgung ausgefallen
9	0	Kein BUSOFF aufgetreten
	1	BUSOFF aufgetreten
10	0	Kein Teilnehmeradressen-Duplizierungsfehler
	1	Teilnehmeradressen-Duplizierungsfehler
11–	0	Reserviert

Sie können die Statusinformationen im DeviceNet-Master abrufen, indem Sie einen Verbindungspfad auswählen, der Statusinformationen enthält. Die Statusinformationen enthalten ein Bit. Bit 2 weist daraufhin, dass die Netzwerkspannung unter den Pegel abgefallen ist, der in den TJ1-DRT DeviceNet-Parametern eingestellt wurde. Sie können die TJ1-DRT DeviceNet-Parameter mit einem DeviceNet-Konfigurator einstellen. Die Standardeinstellung ist 11 V.

4.6 MECHATROLINKII

Das MECHATROLINK-II-Protokoll ist ein serieller Bus, der Bewegungen auf deterministische Weise steuert.

Die Anzahl der MECHATROLINK-II-Geräte wird durch die Zykluszeit für den Datenaustausch festgelegt:

- Für 1 bis 4 Geräte kann die Zykluszeit 0,5 ms, 1 ms oder 2 ms betragen.
- Für 5 bis 8 Geräte kann die Zykluszeit 1 ms oder 2 ms betragen.
- Für 9 bis 16 Geräte beträgt die Zykluszeit 2 ms.

Die zyklische Übertragung hat zwei Phasen:

- Der TJ1-ML__ sendet den Sollwertbefehl an die MECHATROLINK-Slaves.
- Die Slaves senden eine Rückmeldung und Statusinformationen an den TJ1-ML__.

Das MECHATROLINK-II-System verwendet eine Synchronisationsuhr und Rundrufmeldungen, um sicherzustellen, dass alle Slaves die Befehle zur gleichen Zeit ausführen.

Außerdem werden andere Daten mit einer geringeren Geschwindigkeit übertragen, z. B. das Lesen und Schreiben von Parametern.

Es gibt spezifische BASIC-Befehle, mit denen MECHATROLINK-Slave-Baugruppen direkt angesprochen werden können.

- **DRIVE_CLEAR** Dieser Befehl setzt mit einer MECHATROLINK-Meldung einen Alarm in einem MECHATROLINK-Servotreiber zurück.
- **OP(45,ON)**: Dieser Befehl setzt in einem dezentralen MECHATROLINK-E/A-Modul einen Ausgang auf EIN.

5 Trajexia Tools-Schnittstelle

5.1 Einführung

Trajexia Tools ist das Software-Tool zur Programmierung des Trajexia-Systems. Diese Software erlaubt es dem Anwendungsingenieur, Trajexia-Projekte zu verwalten und Programme zu editieren. Sie enthält einige nützliche Werkzeuge, die später in diesem Kapitel beschrieben werden, z. B. Ausführen/Stoppen/Schrittweises Ausführen einzelner Programme, Hinzufügen von Haltepunkten, Ausführung direkter Befehle zum Lesen/Schreiben von Variablen, Oszilloskop-Funktionen und Programmierung der Servotreiber.
Die Verbindung zum TJ1-MC__ erfolgt über Ethernet.
Die Kommunikationseinstellungen müssen vor dem Anschließen einer Baugruppe vorgenommen werden.
Das Trajexia Tools-Software-Tool ist darauf ausgelegt, online mit einem TJ1-MC__ zusammenzuarbeiten.
Trajexia Tools besteht aus folgenden Elementen:

- Dem Software-Tool für den TJ1-MC__ (Motion Perfect 2)
- CX-Server
- CX-Drive zur Programmierung und Einrichtung der Servotreiber und Frequenzumrichter.

Trajexia Tools kann dazu verwendet werden, andere OMRON-Motion Controller über serielle Kommunikation zu programmieren:
C200HW-MC402E, R88A-MCW151-E and R88A-MCW151-DRT-E.

5.2 Spezifikationen und Anschlüsse

5.2.1 PC-Spezifikationen

Die PC-Spezifikationen zur Verwendung von Trajexia Tools sind:

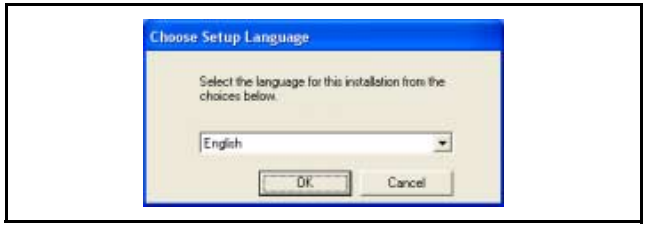
Beschreibung	Mindestspezifikation	Empfohlene Spezifikation
CPU	Pentium, 300 MHz	Pentium, 1 GHz
RAM	64 MB	256 MB
Festplattenspeicher	140 MB	140 MB
Betriebssystem	Windows TM 98	Windows XP
Anzeige	800 x 600 256 Farben	1024 x 768 24-Bit-Farbe
Kommunikation	Ethernet 10BaseT	Ethernet 100BaseT
Internet	Explorer V5.0	Explorer V6.0

Verwenden Sie die neueste Version von Trajexia Tools. Updates erhalten Sie bei Ihrem OMRON-Händler. Die Software ist auch über die Trajexia-Website erhältlich: www.trajexia.com.

5.2.2 Installieren der Trajexia-Tool-Software

1. Legen Sie die Trajexia-Tools-CD in das CD-ROM-Laufwerk des PCs ein.
2. Das Trajexia-Tools-Setup-Programm wird automatisch gestartet.
3. Wenn das Trajexia-Tools-Setup-Programm nicht automatisch gestartet wird, starten Sie es manuell: Führen Sie im Stammverzeichnis der CD die Datei **setup.exe** aus.
4. Wählen Sie die gewünschte Sprache aus der Dropdown-Liste. Klicken Sie auf **OK**.

Abb.1



5. Das Trajexia-Tools-Setup-Fenster wird angezeigt. Klicken Sie auf **Weiter**.

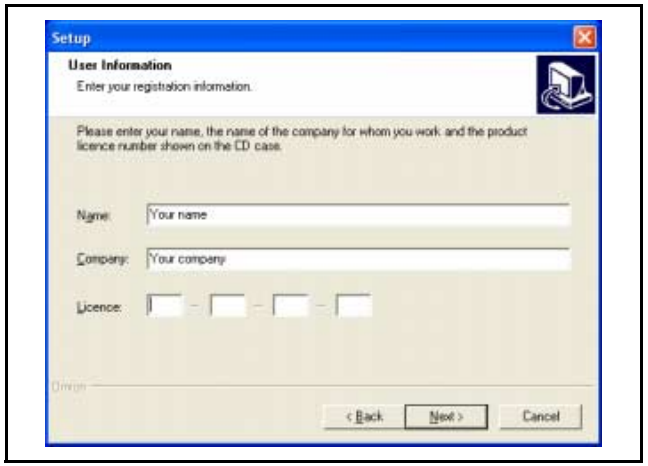
Abb.2

6. Klicken Sie auf **Ja**, um der Lizenzvereinbarung zuzustimmen, und setzen Sie den Vorgang fort.

Abb.3

7. Geben Sie Ihren Namen in das Feld **Name** ein.
8. Geben Sie den Namen Ihres Unternehmens in das Feld **Firma** ein.
9. Geben Sie die Nummer Ihrer Benutzerlizenz in die Felder **Lizenz** ein. Die Nummer Ihrer Benutzerlizenz befindet sich auf dem Etikett, das an der Hülle der Trajexia-Tools-CD angebracht ist.
10. Klicken Sie auf **Weiter**.

Abb.4

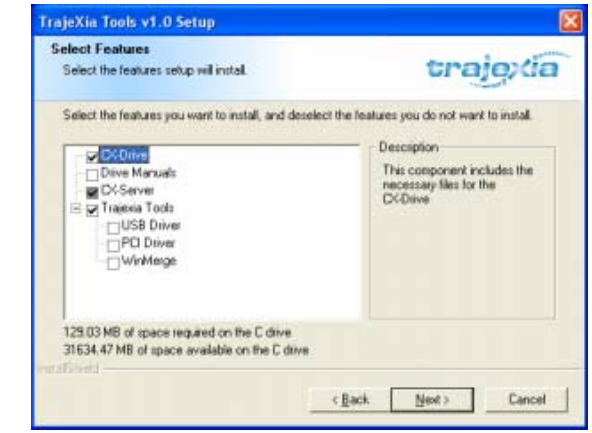


11. Klicken Sie auf **Ja**.

Abb.5

12. Klicken Sie auf **Weiter**.

Abb.6

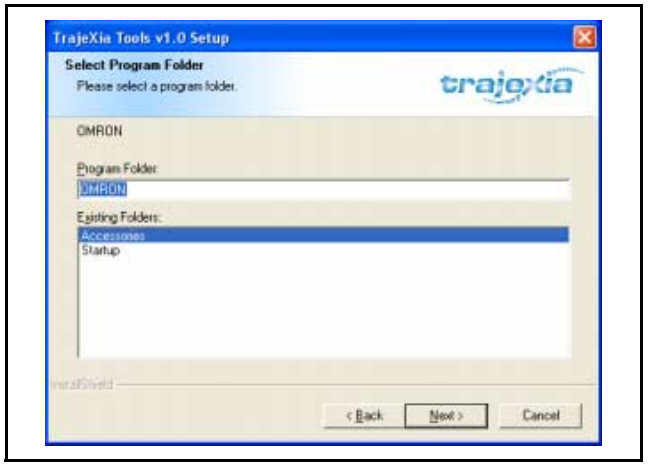


13. Klicken Sie auf **Weiter**.

Abb.7

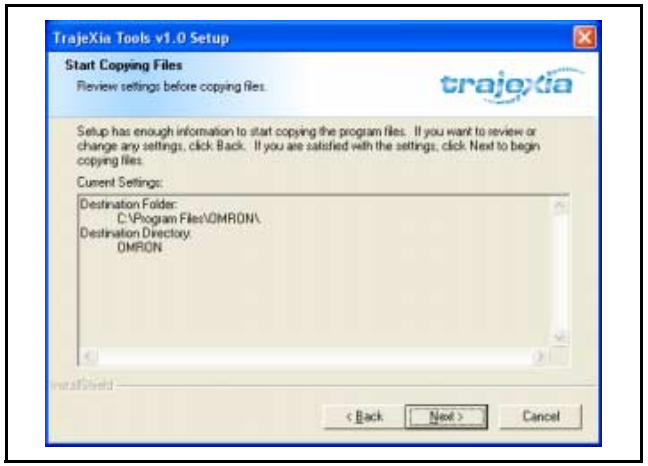
14. Klicken Sie auf **Weiter**.

Abb.8



15. Klicken Sie auf **Weiter**.
16. Das Trajexia-Tools-Setup-Programm kopiert die Dateien auf Ihren PC. Dieser Vorgang kann einige Minuten dauern.

Abb.9



17. Klicken Sie auf **Fertigstellen**. Das Fenster für die CX-Drive Readme-Datei wird geöffnet. Schließen Sie dieses Fenster.

Abb.10

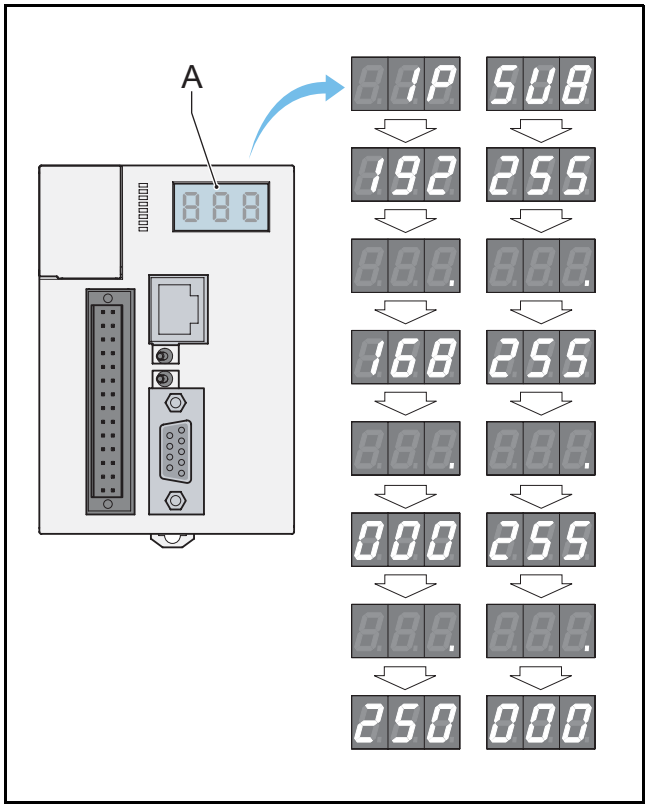
5.2.3 Anschließen an den TJ1-MC__

Sie benötigen ein Patch- oder Crossover-Ethernet-Kabel, um den PC an den TJ1-MC__ anzuschließen.

Hinweis
Wenn Sie offline arbeiten, kann der Simulator verwendet werden. Mit der Simulation kann Trajexia Tools an einen virtuellen Controller angeschlossen werden. Auf diese Weise kann die Programmierung offline erfolgen. Der „Simulator“ erkennt nicht alle spezifischen Befehle für den TJ1-MC__.

1. Schließen Sie das Trajexia-System an die Stromversorgung an.
2. Wenn die IP-Adresse und die Subnetzmaske des TJ1-MC__ erneut angezeigt werden sollen, schalten Sie das Trajexia-System aus und anschließend wieder ein.
3. Schließen Sie das Ethernet-Kabel an die Ethernet-Schnittstelle des PCs an.
4. Schließen Sie das Ethernet-Kabel an die Ethernet-Schnittstelle des TJ1-MC__ an. Die IP-Adresse des TJ1-MC__ wird viermal in der LED-Anzeige (A) angezeigt.
5. Wenn Sie die Trajexia Tools-Software starten, versucht sie, mit dem Controller zu kommunizieren. Wenn Sie Trajexia Tools zum ersten Mal starten, sind die Kommunikationseinstellungen nicht geeignet; brechen Sie den Vorgang ab (siehe Abb. 12), und nehmen Sie Ihre eigenen Kommunikationseinstellungen vor.

Abb.11



6. Starten Sie das Trajexia-Tools-Programm auf Ihrem PC.
   Wählen Sie im Windows **Start** menü:
- **Programme**
- **OMRON**
- **Trajexia-Tools**
- **Trajexia-Tools**

7. Der **Trajexia Tools**-Startbildschirm wird angezeigt. Warten Sie, bis die Schaltfläche **Abbrechen** angezeigt ist. Klicken Sie anschließend auf **Abbrechen**.

Abb.12

8. Wählen Sie das Menü:
- **Optionen**
- **Kommunikation**

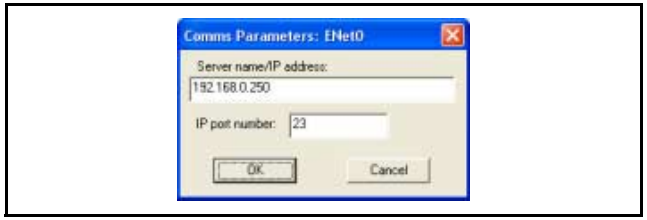
Abb.13

9. Stellen Sie sicher, dass **ENet0** in der Liste ausgewählt ist.
10. Klicken Sie auf **Konfigurieren**.

Abb.14

11. Geben Sie im Feld **Servername/IP-Adresse** den Wert **192.168.0.250** ein.
12. Klicken Sie auf **OK**.

Abb.15



13. Klicken Sie auf **OK**.

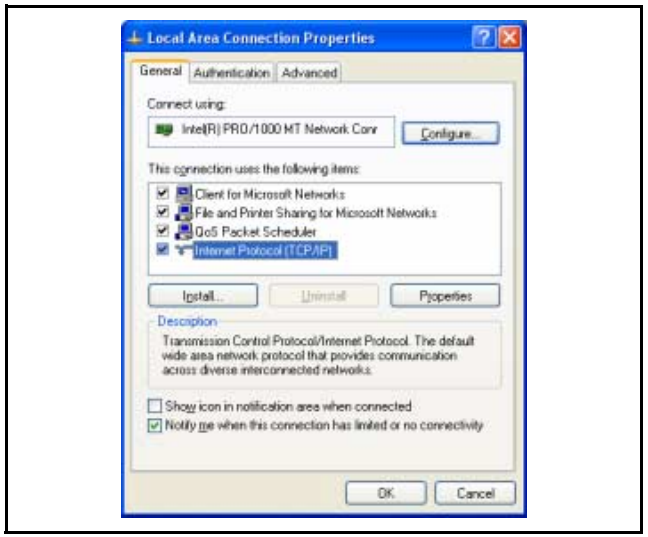
Abb.16

14. Öffnen Sie die Systemsteuerung von Windows auf Ihrem PC.
15. Doppelklicken Sie auf das Symbol **Netzwerkverbindungen**.
16. Klicken Sie mit der rechten Maustaste auf das Symbol **LAN-Verbindung**. Klicken Sie auf das Menü **Eigenschaften**.

Abb.17

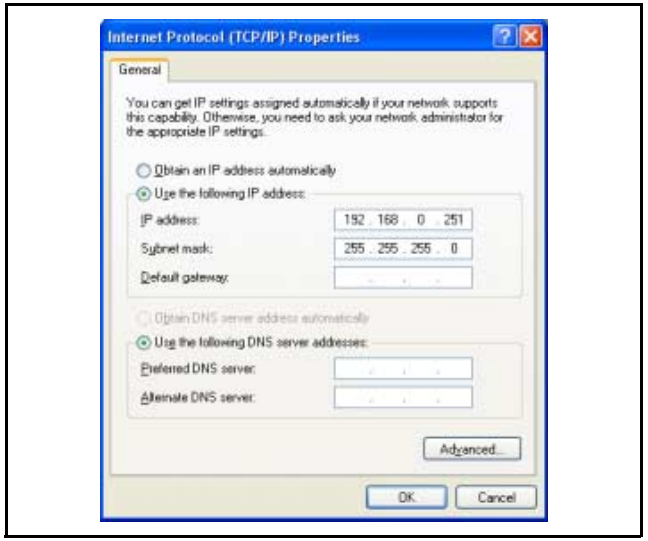
17. Klicken Sie auf die Registerkarte **Allgemein**.
18. Wählen Sie **Internetprotokoll (TCP/IP)** aus der Liste.
19. Klicken Sie auf **Eigenschaften**.

Abb.18



20. Klicken Sie auf die Registerkarte **Allgemein**.
21. Wählen Sie **Folgende IP-Adresse verwenden**.
22. Geben Sie im Feld **IP-Adresse 192.168.0.251** ein.
23. Geben Sie im Feld **Subnetzmaske 255.255.255.0** ein.
24. Klicken Sie auf **OK**.
25. Klicken Sie auf **OK**.
26. Schließen Sie den Bildschirm „Netzwerkverbindungen“.

Abb.19

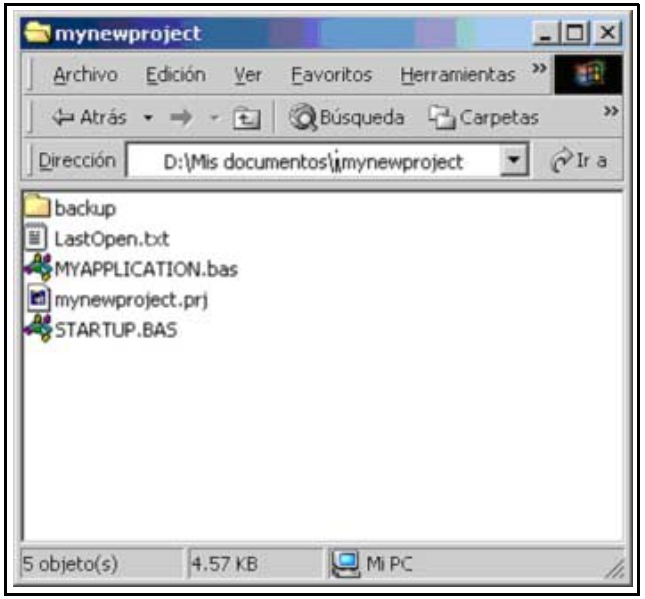


5.3 Projekte

5.3.1 Trajexia Tools Projects

Projects erleichtert den Aufbau- und Entwicklungsprozess einer Anwendung. Auf dem PC ist eine Festplattenkopie aller Programme, Parameter und Daten verfügbar, die zur Programmierung des Systems verwendet werden. Der Benutzer definiert ein Projekt, Trajexia Tools hält die Konsistenz zwischen dem Projekt auf dem PC und dem Trajexia-System aufrecht. Bearbeitete Programme werden automatisch als Duplikat auf dem PC gespeichert.

Abb.20



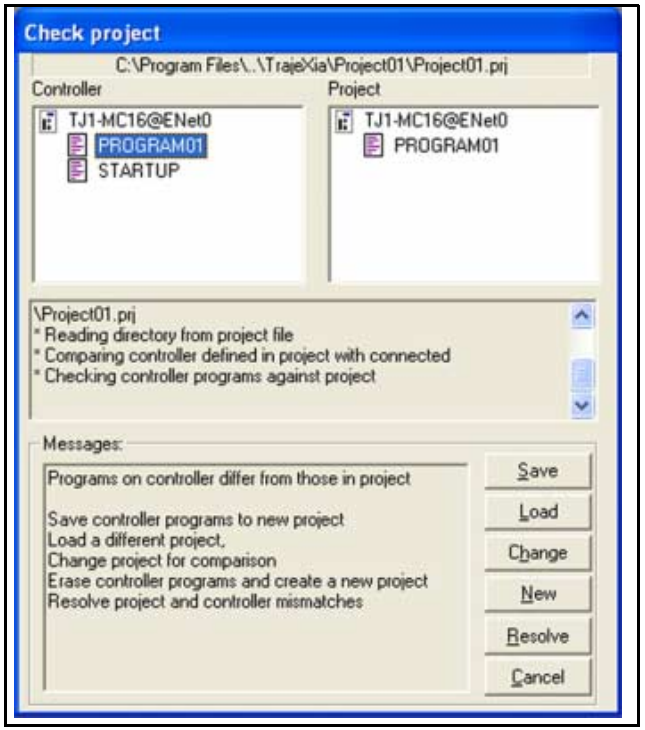
Ein Trajexia-Tools-Projekt besteht aus einer **project.prj**-Datei, die die Trajexia-Konfiguration, die Servotreiber-Parameter und eine Reihe von **.bas**-Programmen enthält. Diese Dateien werden in einem Ordner mit demselben Namen oder der **.prj**-Datei gespeichert.

Hinweis:
Ein Programm, das auf einem Computer erstellt wurde, kann auf einem anderen nicht ordnungsgemäß geöffnet werden. Um dieses Problem zu vermeiden, kopieren Sie das gesamte Projektverzeichnis von einem Computer zum anderen.

5.3.2 Das Fenster „Projekt überprüfen“

Trajexia Tools startet das Fenster „Projekt überprüfen“, wenn eine Verbindung zum Trajexia-System besteht. Es wird ein Vergleich zwischen den Programmdateien im Trajexia-System und den Programmdateien auf dem PC durchgeführt.

Abb.21



Wenn die Programmdateien unterschiedlich sind, wird im Fenster „Projekt überprüfen“ Folgendes angezeigt:

- **Speichern**
- **Laden**
- **Ändern**
- **Neu**
- **Auflösen**
- **Abbrechen**

Speichern

Lädt das Projekt im Trajexia-System auf den PC.
Ein Projekt gleichen Namens auf dem PC wird überschrieben.
Bevor Sie das Projekt auf dem PC speichern, legen Sie von dem Programm auf dem PC eine Sicherungskopie an.

Laden

Lädt das Projekt im PC zum Trajexia-System herunter.
Das Projekt im Trajexia-System wird überschrieben. Bevor Sie das Projekt in das Trajexia-System laden, legen Sie von dem Programm im Trajexia-System eine Sicherungskopie an.

Ändern

Erlaubt es, ein Projekt auf dem PC zu öffnen, das sich von dem Ausgangsprojekt unterscheidet.
Wenn das Projekt im Trajexia-System und auf dem PC nicht gleich sind, können Sie mit **Ändern** ein anderes Projekt auf dem PC auswählen. Trajexia Tools überprüft wieder beide Projekte.
Diese Funktion ist erforderlich, wenn an mehreren Anwendungen bei unterschiedlichen Projekten gearbeitet wird.

Neu
Löscht das Projekt im Trajexia-System und startet ein neues Projekt auf dem PC. Trajexia Tools legt ein neues Verzeichnis mit dem Namen des Projektes an, das die neue Projektdatei enthält. Der Name des Verzeichnisses muss mit dem Namen des Projektes identisch sein, sonst kann das Projekt nicht geöffnet werden.

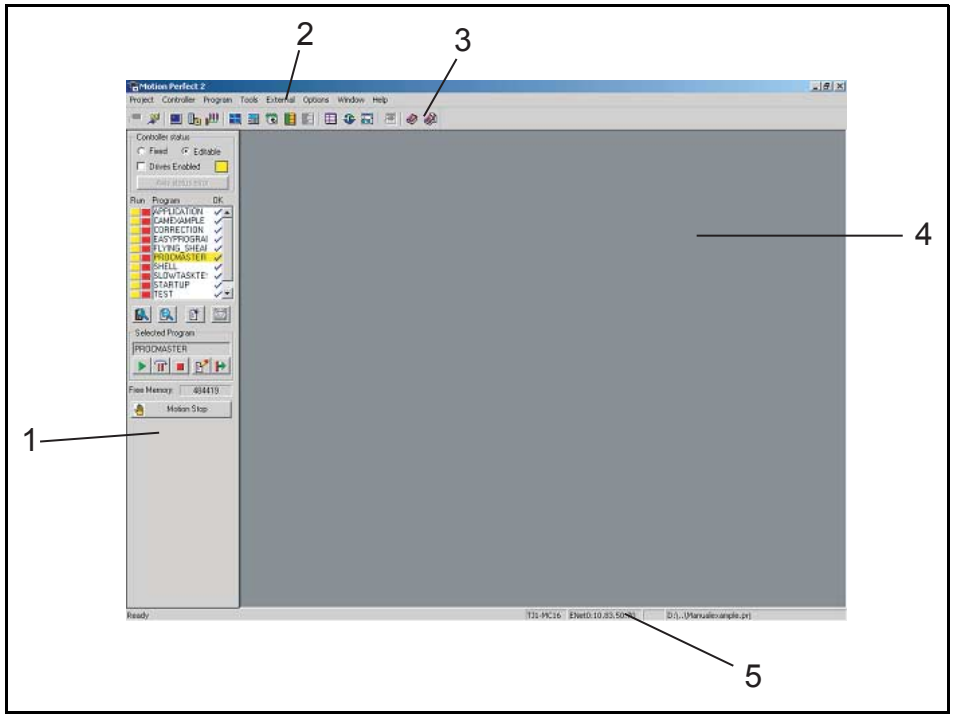
Auflösen
Vergleicht das Projekt in Trajexia Tools mit dem Projekt auf dem PC. Diese Option bietet die Möglichkeit, die Unterschiede gesondert für jedes einzelne Programm innerhalb des Projektes zu **Speichern**, **Laden** oder **Untersuchen**. Diese Option erlaubt eine Offline-Modifizierung eines Programms mit dem Simulator und einen Download dieses Programms zum TJ1-MC__. Diese Option ermöglicht es auch, dass mehrere Personen gleichzeitig an demselben Projekt arbeiten.

Abbrechen
Stoppt den Verbindungsprozess. Trajexia Tools startet in getrenntem Zustand.

5.4 Trajexia-Tools-Anwendungsfenster

Das Trajexia-Tools-Anwendungsfenster besteht aus folgenden Teilen:
1. Bedienfeld
2. Menüleiste
3. Werkzeugleiste
4. Arbeitsplatz
5. Statusleiste

Abb.22



5.4.1 Bedienfeld

Das Bedienfeld erlaubt schnellen und einfachen Zugriff auf die am häufigsten verwendeten Steuerfunktionen zur Verwaltung und Inbetriebnahme eines Projekts.

5.4.2 Menüleiste

Die Menüleiste besteht aus folgenden Elementen:
- **Projekt**
- **Steuerung**
- **Programm**
- **Tools**
- **Extern**
- **Optionen**
- **Fenster**
- **Hilfe**

Die Menüs sind detailliert in Abschnitt „Menübeschreibungen“ (S. 196) beschrieben.

5.4.3 Werkzeugleiste

Verbinden Verbindet Trajexia Tools mit dem Trajexia-System. Siehe „Verbinden“ im Abschnitt „Steuerungsmenü“ (S. 198).

Trennen Trennt die Verbindung von Trajexia Tools zum Trajexia-System. Siehe „Trennen“ im Abschnitt „Steuerungsmenü“ (S. 198).

Terminal Befehlszeilen-Editor. Siehe „Terminal“ im Abschnitt „Tools-Menü“ (S. 205).

Achsenparameter Siehe „Achsenparameter“ im Abschnitt „Tools-Menü“ (S. 205).

Intelligente Antriebe Siehe „Intelligente Antriebe“ im Abschnitt „Tools-Menü“ (S. 205).

Oszilloskop Das Software-Oszilloskop kann zur Rückverfolgung von Achsen- und Bewegungsparametern verwendet werden. Dies hilft bei der Programmentwicklung und der Inbetriebnahme des Systems. Siehe „Oszilloskop“ im Abschnitt „Tools-Menü“ (S. 205).

Tastenfeld In Trajexia nicht implementiert.

Achse bewegen Dieses Fenster erlaubt es dem Benutzer, die Achsen im Trajexia-System manuell zu bewegen. Siehe „Achse bewegen“ im Abschnitt „Tools-Menü“ (S. 205).

Digitale E/A Siehe „Status digitale E/A“ im Abschnitt „Tools-Menü“ (S. 205).

Analogeingang Siehe „Analogeingang“ im Abschnitt „Tools-Menü“ (S. 205).

TABLE-Werte Siehe „TABLE- und VR-Werte“ im Abschnitt „Tools-Menü“ (S. 205).

VR-Werte Siehe „TABLE- und VR-Werte“ im Abschnitt „Tools-Menü“ (S. 205).

Variablen beobachten In Trajexia nicht implementiert.

Simulator In Trajexia nicht vollständig implementiert.

Trajexia-Tools-Hilfe Öffnet die Trajexia-Tools-Hilfe.

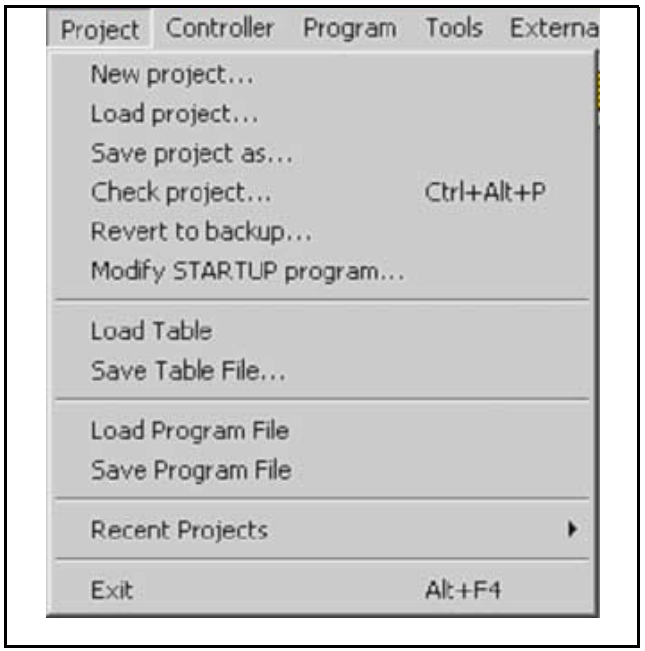
Trio-BASIC-Hilfe Öffnet die Trio-BASIC-Hilfe.

5.5 Menübeschreibungen

5.5.1 Projektmenü

Mit dem **Projekt**-Menü können Sie Trajexia-Tools-Projekte erstellen, laden und speichern.

Abb.23



Neues Projekt
Löscht das Projekt im Trajexia-System und startet ein neues Projekt auf dem PC. Trajexia Tools legt ein neues Verzeichnis mit dem Namen des Projektes an, das die neue Projektdatei enthält. Der Name des Verzeichnisses muss mit dem Namen des Projektes identisch sein, sonst kann das Projekt nicht geöffnet werden.

Projekt laden
Öffnet ein Projekt auf dem PC. Trajexia Tools lädt das Projekt im PC zum Trajexia-System herunter.
Das Projekt im Trajexia-System wird überschrieben. Bevor Sie das Projekt in das Trajexia-System laden, legen Sie von dem Programm im Trajexia-System eine Sicherungskopie an.

Projekt speichern unter...
Lädt das Projekt im Trajexia-System zum PC herauf und speichert es in einem Verzeichnis auf der Festplatte des PCs.

Projekt überprüfen
Es wird eine Überprüfung zwischen dem Projekt im Trajexia-System und dem auf dem PC durchgeführt. Die Prüfsummen und Programminhalte werden verglichen.

Zur Sicherungskopie zurückkehren...
Wenn Trajexia Tools mit dem MC16 verbunden ist, vergleicht das Programm das Projekt im Controller mit dem auf dem PC und legt eine Sicherungskopie an. **Die Option „Zur Sicherungskopie zurückkehren“** kann verwendet werden, wenn Sie alle an dem Projekt und den BASIC-Programmen vorgenommenen Modifikationen rückgängig machen wollen, während eine Verbindung zum Controller besteht.

Dabei werden die BASIC-Programme zu den im Sicherungsverzeichnis gespeicherten Versionen zurückgeändert.

START-Programm modifizieren

Das **Start**-Programm überprüft die Anzahl der Teilnehmer in einem an das System angeschlossenen MECHATROLINK-II-System. Verwenden Sie das Programm **STARTUP modifizieren**, um ein Startprogramm zu ändern, das über das Fenster **Intelligente Antriebe** erstellt ist.

Abb.24

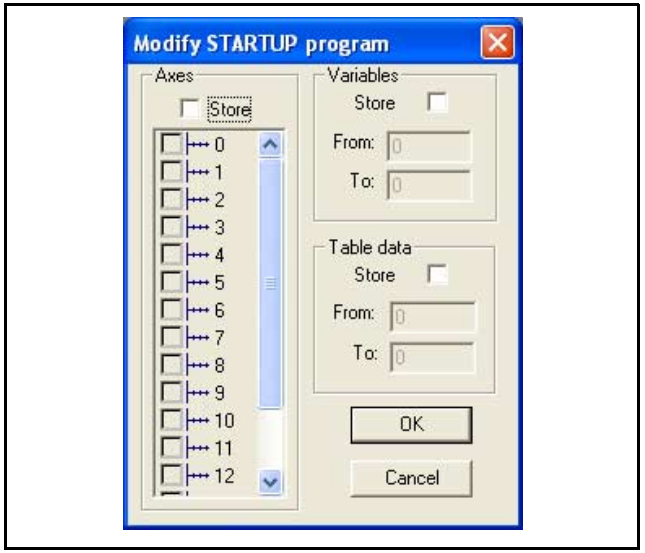


Tabelle laden

Eine Liste mit Tabellenwerten kann aus einer externen Datei mit der Endung ***.lst** oder ***.bas** geladen werden. Die Werte werden importiert und in TABLE-Werten gespeichert.

Tabellendatei speichern...

Speichert eine ***.lst**- oder ***.bas**-Datei aus TABLE-Werten im Projektverzeichnis.

Programmdatei laden

Lädt eine Datei, die Code enthält, der in einem Task ausgeführt werden kann.

Programmdatei speichern

Speichert die Programmdatei im **.txt**-Format.

Kürzliche Projekte

Sie können die Projekte öffnen, die kürzlich mit der Trajexia Tools-Software bearbeitet worden sind.

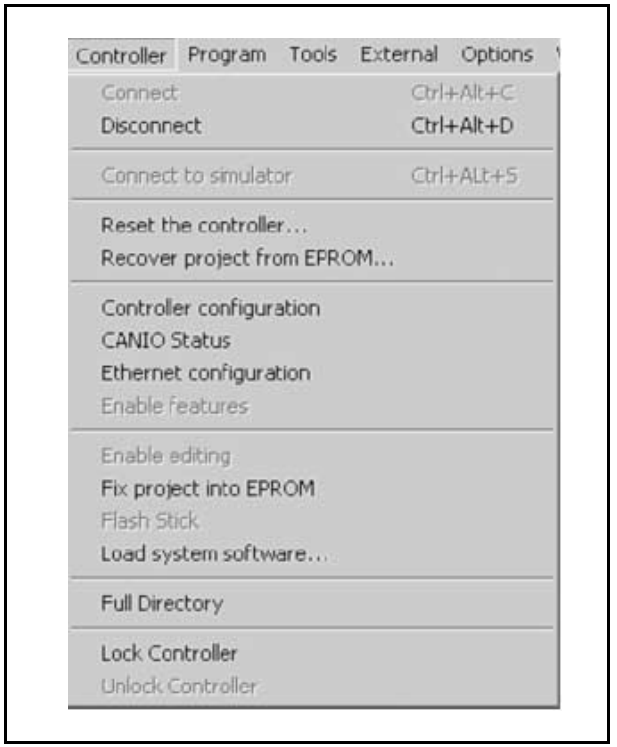
Beenden

Schließt die Trajexia Tools-Anwendung.

5.5.2 Steuerungsmenü

Mit dem Steuerungsmenü können Sie die Kommunikation zwischen dem PC und dem Trajexia-System einstellen und das Trajexia-System steuern.

Abb.25



Verbinden
Stellt eine Verbindung zum Trajexia-System her und startet den Projektmanager. Dieser Befehl ist verfügbar, wenn Trajexia Tools vom System getrennt ist.

Trennen
Trennt die Verbindung zum Trajexia-System. Dieser Befehl ist verfügbar, wenn Trajexia Tools mit dem System verbunden ist.

Mit Simulator verbinden
In Trajexia nicht vollständig implementiert.

Controller zurücksetzen
Führt eine Software-Rücksetzung des Trajexia-Systems durch. Die Verbindung von Trajexia Tools mit dem Trajexia-System wird getrennt.

Projekt aus EPROM wiederherstellen
Setzt das Trajexia-System zurück und stellt die Programme, die im EPROM gespeichert sind, auf dem PC wieder her.

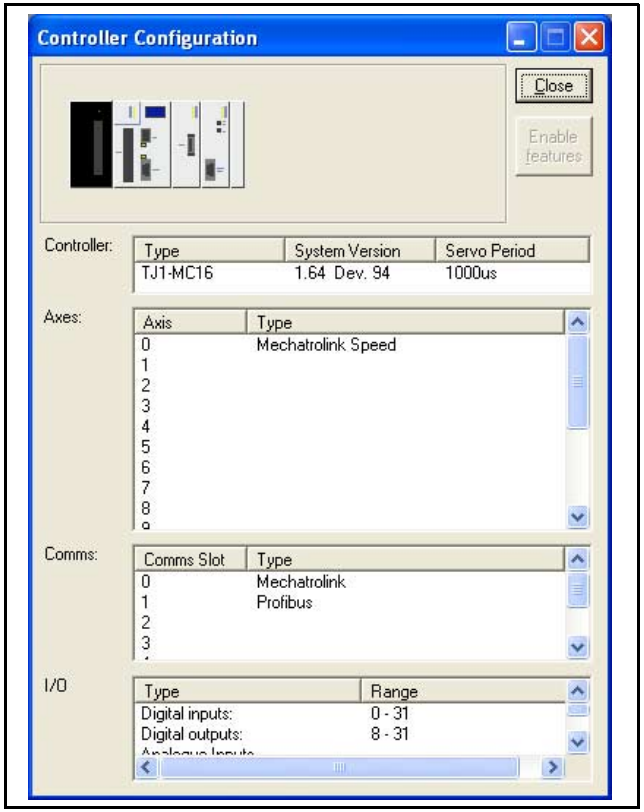
Controller-Konfiguration

Zeigt den Hardware-Konfigurationsbildschirm der Controller-Hardware an, die an den PC angeschlossen ist.
Controller: Der PC ist an einen Trajexia Motion-Controller (TJ1-MC__) mit 1,66 Dev. 27-Software angeschlossen. Die Servo Zykluszeit ist 1000 µs.
Achse: Zeigt die verfügbaren Achsen an.
Komm: Die Kommunikationsfunktionen des Trajexia Motion-Controllers.
E/A: Typ und Bereich der digitalen, analogen und virtuellen Eingänge und Ausgänge.

CANIO

In Trajexia nicht implementiert.

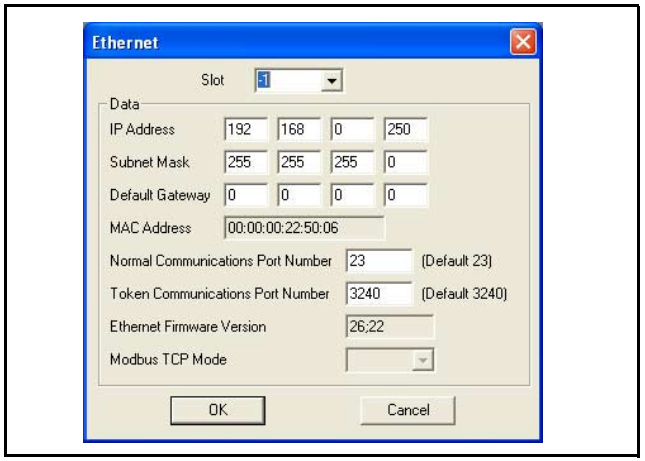
Abb.26



Ethernet-Konfiguration

Erlaubt eine Änderung der Ethernet-Konfiguration und IP-Adresse der Controller-Hardware.
Slot: Für Trajexia immer –1.
IP-Adresse: Die IP-Adresse des Trajexia Motion-Controllers. Sie ist nicht die gleiche wie die IP-Adresse des PCs.
Subnetzmaske: Die Subnetzmaske für den Trajexia Motion-Controller und den PC müssen identisch sein.
Standard-Gateway: Ein Teilnehmer im Netzwerk, der als Zugang zu einem anderen Netzwerk dient. Dies ist nur erforderlich, wenn Trajexia mit einem Gerät in einem anderen Subnetz kommunizieren soll.
MAC-Adresse: Media Access Control-Adresse, eine Hardware-Adresse, durch die jeder Teilnehmer im Netzwerk eindeutig identifiziert wird. Diese Adresse ist schreibgeschützt.
Normale Kommunikationsschnittstellen-Nummer: Die TCP-Schnittstelle, die zur Kommunikation mit Trajexia Tools verwendet wird.
Token-Kommunikationsschnittstellen-Nummer: Die TCP-Schnittstelle, die zur Kommunikation mit der Motion ActiveX-Steuerung des PCs verwendet wird.

Abb.27



Funktionen aktivieren

In Trajexia nicht implementiert.

Bearbeitung aktivieren

Setzt Trajexia in eine Betriebsart, in der mit der RAM-Version der Programme gearbeitet wird. In dieser Betriebsart können die Programme bearbeitet werden.
Durch die Sicherungsbatterie hält das Trajexia-System die Programme (und globalen Variablen) im RAM gespeichert.
Mit dieser Option wird intern der Parameter **POWER_UP** geändert.
Wenn **POWER_UP** auf 0 gesetzt wird, verwendet Trajexia beim Hochfahren die Programme, die durch die Sicherungsbatterie im RAM gespeichert sind, auch wenn die Programme im Flash-Speicher vorhanden sind. Diese Funktion ist nur verfügbar, wenn **POWER_UP=1**.

Projekt in EPROM fixieren

Kopiert die im Controller gespeicherten Programme in das Flash-EPROM. Alle aktuell im EPROM gespeicherten Programme werden überschrieben. Durch diese Funktion wird **POWER_UP** auf 1 geändert, der RAM wird nach dem Hochfahren mit dem Inhalt des EPROMs überschrieben. In dieser Betriebsart können die Programme nicht bearbeitet werden. Diese Funktion ist nur verfügbar, wenn **POWER_UP=0**.

Systemsoftware laden

Trajexia hat ein Flash-EPROM, in dem die Anwenderprogramme und die System-Software gespeichert sind. Verwenden Sie „System-Software laden“, um ein Upgrade der System-Software auf eine neuere Version durchzuführen.
Es wird ein Dialogfeld mit der Frage angezeigt, ob Sie eine Sicherungskopie des Projektes anlegen wollen und ob der Vorgang fortgesetzt werden soll.
Es wird ein standardmäßiges Dateiauswahlfeld angezeigt. Wählen Sie die benötigte Datei aus.

OMRON empfiehlt, eine neuere Version der System-Software nur dann zu laden, wenn Sie von Ihrem Händler oder von OMRON dazu aufgefordert werden.

Achtung
Laden Sie keine Software, die nicht für den Trajexia Motion-Controller spezifiziert ist. Laden Sie nur Versionen, die spezifisch zur Verwendung mit Trajexia ausgelegt sind. Alle anderen Versionen funktionieren nicht.

Es wird ein Windows-Dialogfeld mit der Frage angezeigt, ob der Vorgang fortgesetzt werden soll.
Klicken Sie zum Starten auf OK. Der Flash-EPROM-Prozess dauert ca. 7 Minuten.

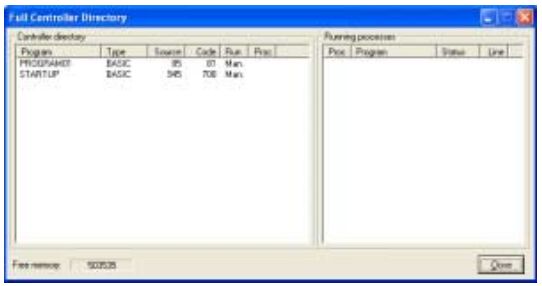
Achtung
Unterbrechen Sie den Software-Upgrade-Prozess nicht. Eine Unterbrechung des Downloads beschädigt die Trajexia-Baugruppe. Wenn Sie die Trajexia-Baugruppe nach dem Flash-EPROM-Prozess nicht wiederherstellen können, wenden Sie sich an Ihre OMRON-Vertretung.

Wenn der Download abgeschlossen ist, wird durch eine Prüfsumme bestätigt, dass der Flash-EPROM-Prozess erfolgreich ist.
Um den Prozess abzuschließen, wählen Sie **Ja** im Dialog-Bestätigungsfenster.
Öffnen Sie das Fenster für die **Controller-Konfiguration**, um die neue Systemversion zu überprüfen.

Vollständiges Verzeichnis

Wird verwendet, um den Inhalt und die Dateistruktur des TJ1-MC__-Verzeichnisses zu überprüfen.

Abb.28



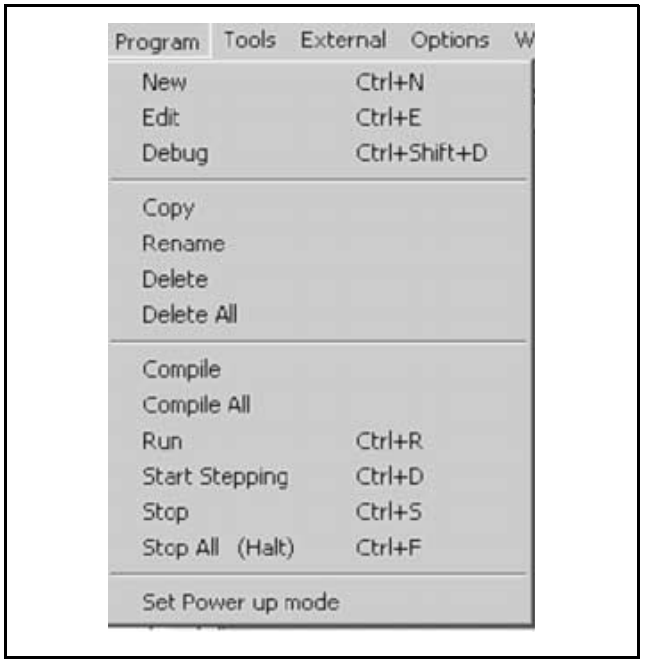
Sperre/Freigabe

Sperrt das Trajexia-System, um unautorisierten Zugriff zu verhindern.
Ist das Trajexia-System gesperrt, ist es nicht möglich, Trajexia-Programme aufzulisten, zu bearbeiten oder zu speichern. Auf das Trajexia-System kann nicht über die Trajexia Tools-Software zugegriffen werden, obwohl die Terminal-und Freigabe-Dialogfelder verfügbar sind.
Zum Sperren des Systems muss eine siebenstellige Zahl eingegeben werden. Verwenden Sie die gleiche Zahl, um das System freizugeben.

5.5.3 Programm-Menü

Das Programm-Menü enthält Menüelemente zur Programmierung eines Trajexia-Projekts.

Abb.29



Neu
Erstellt ein neues Programm.

Bearbeiten
Öffnet ein Projekt zur Bearbeitung.
Sie können den Editor auch über das Bedienfeld starten.
Im Programm-Menü wird zuerst ein Programm-Auswahlfeld angezeigt, in dem Sie aufgefordert werden, die Datei zu bestätigen, die Sie bearbeiten möchten.
Der Trajexia Tools-Editor funktioniert auf ähnliche Weise wie jeder einfache Texteditor auf einem PC. Standard-Operationen wie Blockbearbeitungsfunktionen, Text suchen und ersetzen und Drucken werden unterstützt und entsprechen den standardmäßigen Windows-Tastenkombinationen.
Weitere Funktionen sind BASIC-Syntaxmarkierung, Programmformatierung und Programm-Debugging.

Testen
Überprüft die Syntax eines Programms und gibt mögliche Lösungen vor.
Das Programm wird in einem speziellen Rückverfolgungsmodus geöffnet, in dem jede Zeile einzeln ausgeführt wird. Sie können im Programm Haltepunkte setzen, sodass das Programm ausgeführt wird, bis der Haltepunkt erreicht ist. Die aktuellen Code-Zeile wird im Test-Fenster markiert.
Wird ein Programm im Testmodus ausgeführt, wird jeder geöffnete Editor in den Testmodus gesetzt und dadurch schreibgeschützt.

Kopieren
Kopiert den Inhalt eines Programms in ein anderes Programm.

Umbenennen
Ändert den Namen eines Programms.

Löschen
Löscht das Programm aus der Dateistruktur.

Alle löschen
Löscht alle Programme aus der Dateistruktur.

Kompilieren
Kompiliert das aktuelle Programm im Projekt.

Alle kompilieren
Kompiliert alle Programme im Projekt.

Ausführen
Führt das aktuelle Programm in dem spezifizierten Prozess aus.

Schrittweise Ausführung starten
Führt das aktuelle Programm in dem spezifizierten Prozess schrittweise (Zeile für Zeile) aus.

Stopp
Der Stopp-Befehl stoppt das Programm im TJ1-MC__-Controller. Dies ist nicht das Gleiche wie ein Bewegungsstopp. Das Programm stoppt am Ende des CPU-Zyklus. Die Servomotoren halten die Position.

Alle stoppen
Der Stopp-Befehl stoppt alle Programme im TJ1-MC__-Controller. Dies ist nicht das Gleiche wie ein Bewegungsstopp. Die Programme stoppen am Ende des CPU-Zyklus. Die Servomotoren halten die Position.

Einschaltmodus einstellen

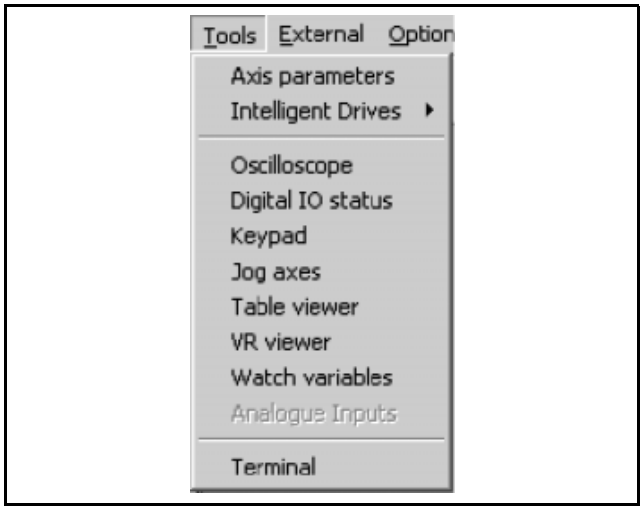
Es ist möglich, die Programme im TJ1-MC__ automatisch zu starten, wenn das System hochfährt. Wählen Sie **Einschaltmodus einstellen**, um das Dialogfeld **Beim Hochfahren ausführen** zu öffnen.
Wählen Sie die das Programm aus, das automatisch ausgeführt werden soll. Rechts im Fenster wird ein kleines Dropdown-Menü angezeigt. Wenn Sie wollen, dass Trajexia den Prozess zuordnet, in dem das Programm laufen soll, wählen Sie „Standard“ als Prozessnummer. Sie können den Prozess auch spezifisch auswählen.

Abb.30

5.5.4 Tools-Menü

Die Werkzeuge von Trajexia Tools können über das Tools-Menü oder die Schaltfläche in der Werkzeugleiste aufgerufen werden.

Abb.31



Achsenparameter

Im Achsenparameter-Fenster können Sie die Bewegungsparameter für jede Achse im Trajexia-System überwachen und ändern.
Das Fenster enthält die Parameter in zwei Blöcken:

- Block 1 (die obere Hälfte des Fensters): enthält Parameter, die durch den Benutzer geändert werden können.
- Block 2 (die untere Hälfte des Fensters): enthält Parameter, die von der Systemsoftware des Trajexia-Systems gesetzt werden, während das System Befehle verarbeitet und den Status externer Eingänge überwacht.

Der Trennbalken, der die zwei Datenblöcke unterteilt, kann mit der Maus verschoben werden.
Wenn der Benutzer einen Baugruppenparameter ändert, werden alle Parameter, die diesen Parameter verwenden, neu gelesen und um diesen Faktor angepasst.
Beispiele für Parametertypen, die durch diesen Parameter beeinflusst werden, sind:

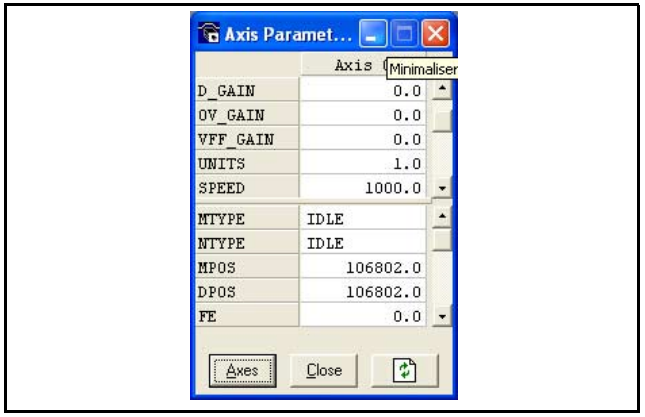
- **SPEED**
- **ACCEL**
- **MPOS**

Der Parameter **Achsenstatus** zeigt den Zustand der Achse an.
Die Farben der Zeichen im Parameterwert zeigen den Status an:

- Grün: Kein Fehler
- Rot: Fehler

Zeichen	Beschreibung
w	Warnung Schleppfehlerbereich
a	Treiber-Kommunikationsfehler
m	Fehler dezentraler Treiber
f	Vorwärts-Grenzwert
r	Rückwärts-Grenzwert
d	Nullpunkteingang

Abb.32



Zeichen	Beschreibung
h	Eingabe Vorschubstopp
e	Schleppfehler
x	Software-Grenzwert Vorwärts
y	Software-Grenzwert-Rückwärts
c	Bewegung wird abgebrochen
o	Drehgeber-Fehler

Die Optionen für das Achsenparameter-Fenster sind:
Achsen: Wählt die Achsen aus, für die Daten angezeigt werden.
Aktualisieren: Um die Belastung für das Trajexia-System zu verringern, werden die Parameter in Block 1 nur dann gelesen, wenn der Bildschirm als Erstes angezeigt oder wenn der Parameterwert gesetzt wird. Wenn ein Parameterwert geändert wird, kann der angezeigte Wert falsch sein. Über die Schaltfläche „Aktualisieren“ erhält Trajexia Tools den Befehl, die komplette Auswahl nochmals zu lesen.

Hinweis:
Wenn Sie einen Parameterwert ändern, müssen Sie die Anzeige aktualisieren, bevor Sie weitere Änderungen vornehmen.

Intelligente Antriebe

„Intelligente Antriebe“ ermöglicht Zugriff auf die Konfiguration und die Startprogramme der einzelnen Antriebe, die konfiguriert werden können.
Das Tool „Intelligente Antriebe“ zeigt die Trajexia-Konfiguration an, wie sie beim Hochfahren erkannt wird. Wenn Sie auf die Baugruppe klicken, wird das nächste Register angezeigt.
Am oberen Fensterrand wird der TJ1-MC__ mit seinen verschiedenen Baugruppen angezeigt. In diesem Beispiel:

- TJ1-MC__ mit Baugruppennummer –1
- TJ1-ML__ mit Baugruppennummer 0
- TJ1-PRT mit Baugruppennummer 1
- TJ1-FL02 mit Baugruppennummer 2.

Wenn mehr als ein TJ1-ML__ im System vorhanden ist, werden mehrere Register angezeigt.
Über die Schaltfläche **STARTUP-Programm modifizieren** wird ein STARTUP-Programm für die erkannte Konfiguration erstellt.
Im Register für den TJ1-ML__ können Sie die Daten für die erkannten MECHATROLINK-II-Slaves ablesen (einschließlich Frequenzumrichter und E/A-Module).
Durch Klick auf die Schaltfläche **Konfig** (nur verfügbar für Servo und Frequenzumrichter) wird das nächste Fenster angezeigt:
Register **Status**:

- **Treiber-ID/Motor-ID/Firmware-Version:** Zeigt Informationen über den Servotreiber & Servomotor an.
- **Antriebsstatus** zeigt den Inhalt des **DRIVE_STATUS**-Wortes für diese Achse an.
- **Antriebs-E/A** zeigt den Inhalt des **DRIVE_INPUTS**-Wortes für diese Achse an.
- **Antrieb löschen** führt den Befehl **DRIVE_CLEAR** (Servotreiber-Alarm löschen) für diese Achse aus.
- **Antrieb zurücksetzen** führt den Befehl **DRIVE_RESET** (Software hochfahren) für diese Achse aus.
- Mit **Antrieb überwachen** werden die Überwachungswerte im Parameter **DRIVE_MONITOR** aktualisiert.

Abb.33

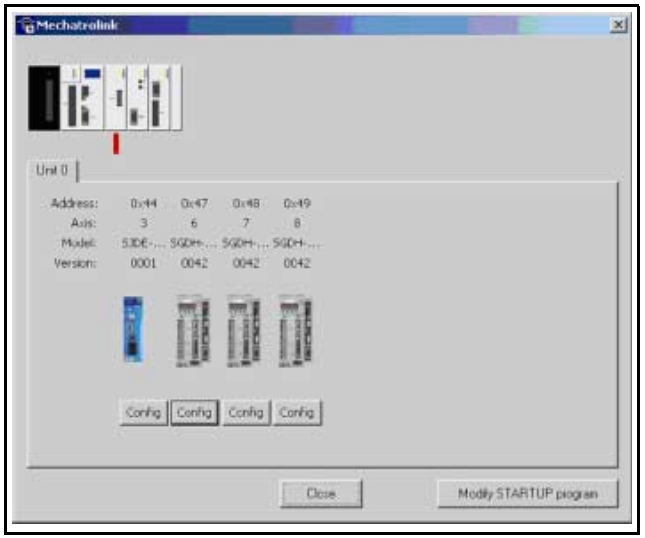
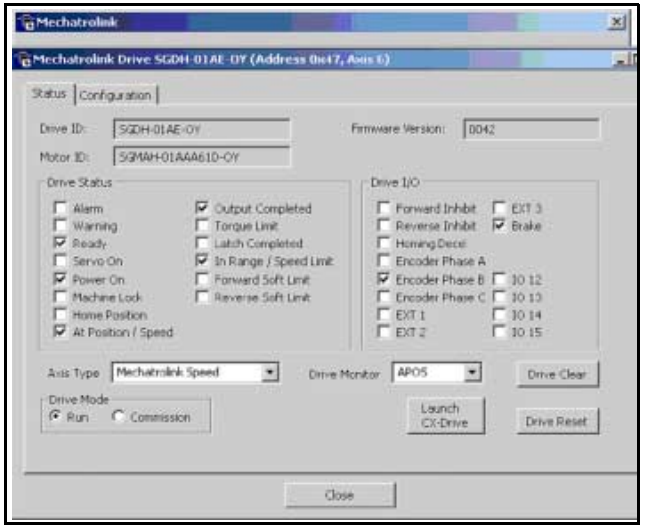


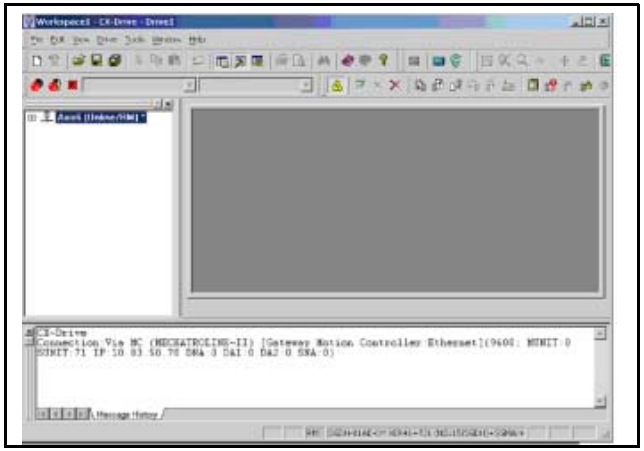
Abb.34



- Mit **Achsentyp** wird der Parameter **ATYPE** für diese Achse ausgewählt. Der Wert wird in das STARTUP-Programm übernommen.
- **Steuerbetriebsart:** Ausführung oder Inbetriebnahme.
  - Wenn die Achse auf „Ausführung“ gesetzt wird, werden der Ausführungsstatus und die Bewegungen vollständig über die Programme gesteuert, die im TJ1-MC__ laufen.
  - Wenn die Achse auf „Inbetriebnahme“ gesetzt wird, werden die Ausführung und die Bewegungen extern über FINS gesteuert; diese Betriebsart ist für die Verwendung der Jog-Funktion und die Einrichtung von CX-Drive implementiert. Dadurch sollen Konflikte mit den Programmen vermieden werden. Während der Inbetriebnahme wird die Achse von den Programmen als virtuelle Achse erkannt.
    Es ist möglich, Parameter aus dem Antrieb gleichzeitig über die Programme oder über FINS zu schreiben und zu lesen, unabhängig von der Betriebsart.
- **CX-Drive starten:** Über Trajexia Tools ist es nur möglich, Parameter eines Servotreibers zu lesen und zu schreiben. Wenn weitere Servotreiber-Funktionen benötigt werden, z. B. Lesen des Alarmcodes, Jog, Einstellen der Steifigkeit oder Autotuning, ist es notwendig, CX-Drive zu starten. Durch Klicken auf diese Schaltfläche wird CX-Drive gestartet, das mit der aktuellen Achse über den TJ1-ML__ verbunden ist.

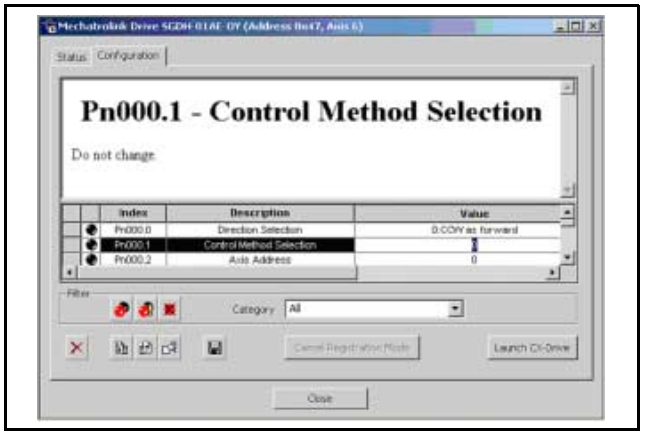
Die einzige Servotreiber-Funktionalität, die von CX-Drive durch MECHATROLINK-II nicht unterstützt wird, ist die Datenaufzeichnungsfunktion, aber stattdessen kann das Trajexia-Tools-Oszilloskop verwendet werden.
Wenn Sie einen Parameter des Servotreibers über CX-Drive ändern, wird dies von Trajexia Tools nicht automatisch registriert. Achten Sie darauf, dass im Servotreiber und im Projekt keine unterschiedlichen Parameter verwendet werden.

Abb.35



Das Register **Konfiguration**.
Das Register **Konfiguration** zeigt ein Parameter-Editorfenster an, das mit dem in CX-Drive identisch ist. Einzelheiten hierzu finden Sie in der Anleitung für CX-Drive.
Die folgende Funktionalität ist neu:
Die Schaltfläche **Speichern**: Hiermit werden die aktuellen Servoparameter im Trajexia-Projekt gespeichert (in der ***.prj**-Datei).
Druckmarkenmodus abbrechen: Um eine schnelle und zuverlässige Antwort zu erhalten, ist es nicht möglich, Parameter zu schreiben, wenn der Druckmarkenmodus im Servotreiber aktiviert ist. Dies ist das Gleiche wie die Ausführung von **REGIST(-1)**.
CX-Drive starten: Die gleiche Funktion wie die Schaltfläche, die unter dem Register „Status“ erläutert ist.

Abb.36

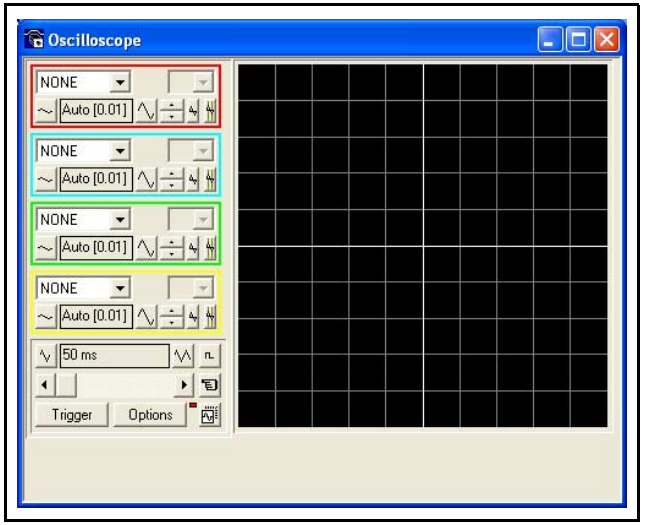


Oszilloskop

Das Software-Oszilloskop kann zur Darstellung von Achsen- und Bewegungsparametern verwendet werden. Dies ist hilfreich bei der Programmentwicklung und der Inbetriebnahme des Systems. Es gibt vier Kanäle, von denen jeder pro **SERVO_PERIOD** bei manuellem Aus- und Einschalten der Spannungsversorgung oder programmbedingter Triggerung 1 Abtastung aufzeichnen kann. Der Controller zeichnet die Daten in der gewählten Häufigkeit auf und lädt sie zur Anzeige zum Oszilloskop herauf. Wenn ein längerer Zeitraum eingestellt ist, werden die Daten abschnittweise erfasst, und die Grafik wird abschnittweise auf dem Display dargestellt. Der Moment, in dem der Controller damit beginnt, die erforderlichen Daten aufzuzeichnen, hängt davon ab, ob er sich in der manuellen oder in der programmgesteuerten Betriebsart befindet.

- **Programm-Betriebsart:** Das Oszilloskop beginnt Daten aufzuzeichnen, wenn ein Auslösebefehl von dem Programm im Controller gesendet wird.
- **Manuelle Betriebsart:** Das Oszilloskop beginnt sofort, Daten aufzuzeichnen.

Abb.37



Oszilloskop-Kanäle

Für alle vier Kanäle des Oszilloskops gibt es eigene Einstellungen. Jede Kanal- Einstellung hat eine farbliche Umrandung, um die Anzeigefarbe für diesen Kanal zu kennzeichnen. Die Einstellungen sind wie folgt:

Parameteranzeige: Die Parameter, die das Oszilloskop aufzeichnen und anzeigen kann, werden aus dem Dropdown-Menü in der oberen linken Ecke jedes Kanal-Kontrollblocks ausgewählt.
Der Parametertyp bestimmt die nächste Kennzeichnung zwischen Achse (Ax) und Kanal (Ch).
Um die Punkte darzustellen, die im TABLE-Speicher des Controllers gespeichert sind, wählen Sie im Dialogfeld „Erweiterte Optionen“ den Parameter TABLE und den Kanal, dessen erste und letzte Punkte konfiguriert sind, aus.
Wenn der Kanal nicht benötigt wird, wählen Sie NONE im Parameter-Listenfeld.
Achsen-/Kanalnummer: Ein Dropdown-Listenfeld zur Auswahl der Achse oder des Kanals für einen Bewegungsparameter oder einen Kanal für einen digitalen oder analogen Eingangs-/Ausgangsparameter.
Y-Bereich nach unten/Y-Bereich nach oben: Die vertikale Skala wird für jeden Kanal ausgewählt und kann für automatische oder manuelle Betriebsart konfiguriert werden. In der automatischen Betriebsart berechnet das Oszilloskop die entsprechende Skala, wenn die Aufzeichnung abgeschlossen ist und bevor das Oszilloskop die Kurve anzeigt. Wenn das Oszilloskop ohne kontinuierliche Triggerung ausgeführt wird, kann es keine geeignete vertikale Skala auswählen. Das Oszilloskop muss gestoppt und neu gestartet werden. In der manuellen Betriebsart wählt der Benutzer die am besten geeignete Skala aus.
Y-Verschiebung: Der Wert für den vertikalen Offset wird dazu verwendet, eine Kurve vertikal in der Anzeige zu verschieben. Diese Funktion ist nützlich, wenn zwei oder mehrere Kurven identisch sind.
Y zurücksetzen: Mit dieser Schaltfläche wird der Y-Verschiebungswert gelöscht.
Aktivieren/Deaktivieren der Cursor-Balken: Wenn das Oszilloskop die Aufzeichnung beendet hat und eine Kurve angezeigt wird, können die Cursor-Balken aktiviert werden. Die Cursor-Balken sind zwei senkrechte Balken in der gleichen Farbe wie die Kanalkurve. Diese markieren die höchsten und niedrigsten Positionspunkte der Kurve. Die durch die Balken dargestellten Werte werden unterhalb des Oszilloskop-Displays angezeigt.
Die Cursor-Balken werden durch Klick auf die Cursor-Schaltfläche aktiviert und deaktiviert. Die Cursor-Balken können ausgewählt und mit dem Maus-Cursor verschoben werden.

Allgemeine Steuerelemente
Die allgemeinen Steuerelemente befinden sich unten links im Oszilloskop-Bildschirm. Die Einstellungen sind wie folgt:
Zeitbasis: Der Zeitbasis-Wert ist der Zeitwert für die einzelnen waagerechten Unterteilungen des Oszilloskops. Die Zeitbasis wird mit den Aufwärts-/Abwärts-Skalierungsschaltflächen an der Seite des aktuellen Feldes für den Zeitbasiswert ausgewählt. Wenn die Zeitbasis größer ist als ein vordefinierter Wert, werden die Daten aus dem Controller abschnittsweise und nicht als durchgängige Datenkurve dargestellt. Die Datenabschnitte werden im Display so dargestellt, wie sie empfangen werden. Der letzte Punkt ist ein weißer Fleck.
X-Verschiebung: Wenn die Aufzeichnung beendet wird, während die Zeitbasis auf einen schnelleren Wert geändert wird, ist nur ein Teil der Kurve in der Anzeige zu sehen. Verwenden Sie die Bildlaufleiste für die X-Verschiebung, um die gesamte Kurve anzeigen zu lassen. Wenn das Oszilloskop so konfiguriert ist, dass es Bewegungsparameter aufzeichnet und Tabellendaten darstellt, kann die Anzahl der im Display dargestellten Punkte durch den Bewegungsparameter ermittelt werden. Weitere Tabellenpunkte, die nicht sichtbar sind, können mit der Bildlaufleiste angezeigt werden. Die Kurve für die Bewegungsparameter kann nicht verschoben werden.
Einzelne/kontinuierliche Triggerung: In der Betriebsart für einzelne Triggerung wird das Oszilloskop ausgeführt, bis die Funktion ausgelöst und ein vom Controller aufgezeichneter Datensatz abgerufen und angezeigt wird.
In der Betriebsart für kontinuierliche Triggerung wird das Oszilloskop ständig ausgeführt und ruft jedes Mal Daten aus dem Controller ab, wenn es getriggert wird und neue Daten aufgezeichnet werden. Das Oszilloskop wird ständig ausgeführt, bis ein zweites Mal auf die Trigger-Schaltfläche geklickt wird.
Triggerung/Datenerfassung stoppen: Durch Klick auf die Trigger-Schaltfläche wird das Oszilloskop aktiviert. In der manuellen Betriebsart beginnt der Controller sofort mit der Datenaufzeichnung. In der Programmbetriebsart wartet er, bis in einem laufenden Programm ein Auslösebefehl gegeben wird. Nachdem auf die Trigger-Schaltfläche geklickt worden ist, ändert sich der Text auf der Schaltfläche zu „Halt“. Wenn sich das Oszilloskop in der Betriebsart für einzelne Aufzeichnung befindet,

wird der Text auf der Schaltfläche zu „Trigger“ zurückgeändert, nachdem die Daten aufgezeichnet und in der Anzeige dargestellt wurden; dadurch wird angezeigt, dass der Vorgang abgeschlossen ist. Das Oszilloskop kann jederzeit gestoppt werden, während es läuft, und die Schaltfläche zeigt den Text „Halt“ an.
Clear configuration (Konfiguration löschen): Die aktuelle Oszilloskop-Konfiguration (der Zustand aller Steuerelemente) wird gespeichert, wenn das Oszilloskop-Fenster geschlossen wird und wiederhergestellt, wenn das Oszilloskop-Fenster das nächste Mal geöffnet wird.
Die Schaltfläche zur Konfigurations-Rücksetzung (sie befindet sich unten rechts im Oszilloskop-Bedienfeld) setzt alle Steuerelemente auf ihre Standardwerte zurück.
Statusanzeige: Die Statusanzeige befindet sich zwischen den Options- und Konfigurations-Rücksetzschaltflächen. Diese Anzeigeleuchte ändert ihre Farbe gemäß dem aktuellen Status des Oszilloskops, und zwar wie folgt:

- Rot: Oszilloskop gestoppt.
- Schwarz: Der Controller wartet darauf, dass das Oszilloskop die Aufzeichnung der angeforderten Daten abschließt.
- Gelb: Daten werden aus dem Controller abgerufen.

Set capture options (Aufzeichnungsoptionen einstellen):
Durch Klick auf diese Optionsschaltfläche wird das Dialogfeld für die erweiterten Konfigurationseinstellungen für das Oszilloskop angezeigt.

Erweiterte Oszilloskop-Optionen

Allgemeine Informationen:

Anzeige von Controller-Tabellenpunkten:
Wenn das Oszilloskop für Tabellen- und Bewegungsparameter konfiguriert ist, richtet sich die Anzahl der auf dem Display dargestellten Punkte nach der Zeitbasis (und den Abtastungen pro Unterteilung). Wenn die Anzahl der darzustellenden Punkte für die Tabellenparameter größer ist als die Anzahl der Punkte für die Bewegungsparameter, erscheinen die zusätzlichen Tabellenpunkte nicht im Display, können aber durch Bewegen der die horizontalen Bildlaufleiste für die Tabellenkurve angezeigt werden. Die Kurve für die Bewegungsparameter kann nicht verschoben werden.

Daten aus dem Controller zum Oszilloskop heraufladen
Wenn die Zeitbasis insgesamt größer ist als ein vordefinierter Wert, werden die Daten aus dem Controller in Blöcken angezeigt; daher wird die Anzeige abschnittweise aktualisiert. Der letzte im aktuellen Abschnitt dargestellte Punkt wird als weißer Fleck angezeigt.
Wenn das Oszilloskop so konfiguriert ist, dass es Bewegungsparameter aufzeichnet und Tabellendaten darstellt, werden die Tabellendaten in einem kompletten Block zurückgelesen; die Bewegungsparameter werden entweder kontinuierlich oder in Blöcken gelesen (je nach Zeitbasis). Auch wenn sich das Oszilloskop in der kontinuierlichen Betriebsart befindet, werden die Tabellendaten nicht neu gelesen; nur die Bewegungsparameter werden kontinuierlich aus dem Controller zurückgelesen.

Aktivieren/Deaktivieren von Oszilloskop-Steuerelementen
Während das Oszilloskop ausgeführt wird, sind alle Steuerelemente des Oszilloskops außer der Auslöse-Schaltfläche deaktiviert. Daher ist es nötig, das Oszilloskop anzuhalten und neu zu starten, wenn die Zeitbasis oder die senkrechte Skala geändert werden soll.

Anzeigegenauigkeit
Der Controller zeichnet die Parameterwerte mit der gewünschten Abtastrate in der Tabelle auf und lädt sie zur Anzeige zum Oszilloskop herauf. Daher ist die angezeigte Kurve in Bezug auf die ausgewählte Zeitbasis genau. Allerdings gibt es eine Verzögerung zwischen dem Zeitpunkt, zu dem die Daten durch den Controller aufgezeichnet werden und dem Zeitpunkt, zu dem sie im Oszilloskop angezeigt werden; diese Verzögerung ergibt sich durch die Zeit, die zum Heraufladen der Daten über die Kommunikationsverbindung benötigt wird.

Samples per division (Abtastungen pro Unterteilung): Das Oszilloskop ist standardmäßig so eingestellt, dass fünf Punkte pro waagrechter (Zeitbasis-) Gitterunterteilung aufgezeichnet werden. Dieser Wert kann mit der angrenzenden Bildlaufleiste angepasst werden. Um die schnellstmögliche Abtastrate zu erreichen, verringern Sie die Anzahl der Abtastungen pro Gitterunterteilung auf 1, und erhöhen Sie die Zeitbasisskala auf den schnellsten Wert (1 Servozyklus pro Gitterunterteilung).

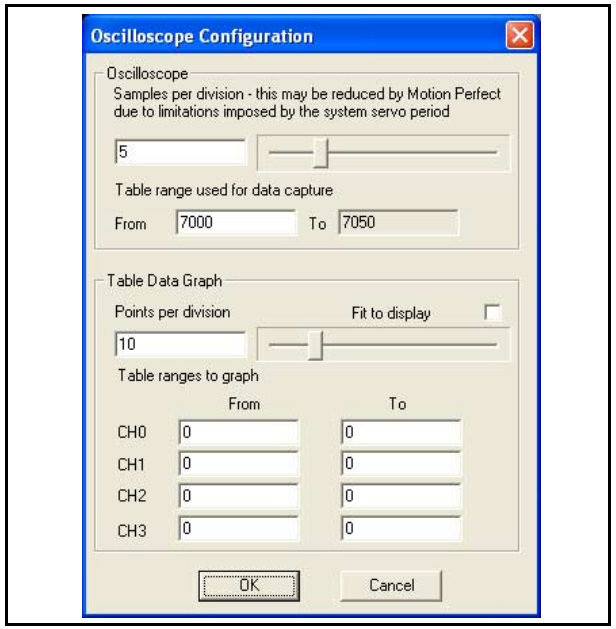
Table range used for data capture (TABLE-Speicherbereich für die Datenerfassung): Der Controller zeichnet die benötigten Parameter-Datenwerte als Tabellendaten auf, bevor diese Werte in das Fenster heraufgeladen werden. Standardmäßig ist der niedrigste verwendete Oszilloskop-Tabellenwert null. Wenn dadurch jedoch ein Konflikt mit Programmen entstehen sollte, die auf dem Controller laufen und die ebenfalls auf diesen Abschnitt der Tabelle zugreifen, kann der untere Tabellenwert geändert werden.
Der obere Oszilloskop-Tabellenwert wird dementsprechend automatisch auf Grund der Anzahl der verwendeten Kanäle und der Anzahl der Abtastungen pro Gitterunterteilung aktualisiert. Wenn Sie als unteren Tabellenwert einen Wert eingeben, der zur Folge hat, dass der obere Tabellenwert den maximal zulässigen Wert für den Controller überschreitet, wird vom Oszilloskop der ursprüngliche Wert verwendet.

Table Data Graph (Tabellendaten-Graph): Es ist möglich, Controller-Tabellenwerte direkt darzustellen; die Tabellengrenzen-Textfelder erlauben es dem Benutzer, bis zu vier Sätze von ersten/letzten Tabellen-Indizes einzugeben.

Parameter-Überprüfungen

Bei der Aufzeichnung von analogen Eingängen ist die schnellste Oszilloskop-Auflösung (Abtastrate) die Anzahl der analogen Kanäle in ms (d. h. 2 analoge Eingänge bedeutet, dass die schnellste Abtastrate 2 ms beträgt). Die Auflösung wird berechnet, indem der Zeitbasis-Skalenwert durch die Anzahl der Abtastungen pro Gitterunterteilung dividiert wird.
Es ist weder möglich, Tabellen-Kanalwerte einzugeben, die über der maximalen TABLE-Größe des Controllers liegen, noch kann ein niedrigerer Oszilloskop-Tabellenwert eingegeben werden. Eine Erhöhung der Abtastungen pro Gitterunterteilung auf einen Wert, der zur Folge hat, dass der obere Oszilloskop-Tabellenwert den maximalen Tabellenwert des Controllers überschreitet, ist ebenfalls nicht zulässig.

Abb.38



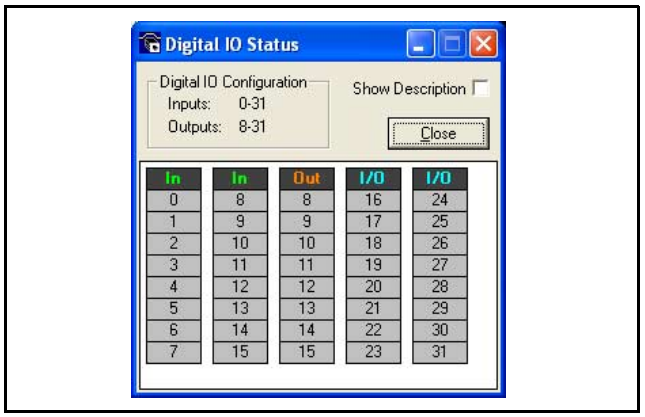
Wenn die Anzahl der Abtastungen pro Gitterunterteilung erhöht und dementsprechend die Zeitbasisskala auf einen schnelleren Wert eingestellt wird, was zu einer unerreichbaren Auflösung führt, setzt das Oszilloskop die Anzahl der Abtastungen pro Gitterunterteilung automatisch zurück.

Status digitale E/A

Dieses Fenster erlaubt es dem Benutzer, den Status aller E/A-Kanäle zu betrachten und die Ausgangskanäle umzuschalten. Optional kann der Benutzer auch eine Beschreibung für jede E/A-Leitung eingeben.
Digital inputs (Digitaleingänge): Die Gesamtanzahl der Eingangskanäle im Trajexia-System.
Digital outputs (Digitalausgänge): Die Gesamtanzahl der Ausgangskanäle im Trajexia-System.
Display: Das Display ist in Blöcke mit acht Anzeigen unterteilt, die Blöcke von acht Eingängen oder Ausgängen darstellen:

- Eingangsblock (In)
  Der Zustand der digitalen Eingänge wird dargestellt. In(0) bis In(15) sind die im TJ1-MC__ integrierten Digitaleingänge. Zusätzliche Digitaleingänge werden im System automatisch zugewiesen, beginnend bei In(32).
- Ausgangsblock (Out)
  Der Zustand der digitalen Ausgänge wird dargestellt. OP(8) bis OP(15) sind die im TJ1-MC__ integrierten Digitalausgänge. Zusätzliche Digitalausgänge werden im System automatisch zugewiesen, beginnend bei OP(32).
- Eingangs-/Ausgangsblöcke (E/A)
  Sie stellen die virtuellen E/A dar, die Sie innerhalb des Programms als Benutzer-Merker verwenden können. Wird einer diese virtuellen Ausgänge eingestellt, wird der entsprechende virtuelle Eingang ebenfalls eingestellt. Eine graue Anzeige bedeutet, dass der entsprechende Eingang oder Ausgang ausgeschaltet ist. Eine farbige Anzeige (gelb, grün, orange, rot, cyan- oder magentafarben) bedeutet, dass der entsprechende Eingang oder Ausgang eingeschaltet ist. Es werden verschiedene Farben verwendet, um verschiedene Typen von Eingängen und Ausgängen darzustellen. Durch Klick auf eine Anzeige, die einen Ausgang (oder die Verknüpfung von Eingang und Ausgang) darstellt, wird der Zustand dieses Ausgangs geändert. Ein Klick auf eine Anzeige, die einen Eingang darstellt, hat keine Auswirkungen.

Abb.39



- Einige Ausgangsschaltkreise erfordern eine externe Stromquelle. In diesem Fall wird der Zustand der intern verknüpften E/A nicht richtig angezeigt, wenn die externe Spannungsversorgung nicht vorhanden ist, weil sich der Eingangszustand nicht ändert, auch wenn ein Ausgang eingeschaltet ist. Ebenso verhält es sich, wenn für einen Ausgang auf Grund eines Fehlers oder einer Überlastung eine Strombegrenzung geschaltet wird.
- Show description (Beschreibung anzeigen): Durch Aktivieren des Kontrollkästchens für die Option „Show description“ (Beschreibung anzeigen) kann die Anzeige der Beschreibungen ein- oder ausgeschaltet werden. Die Beschreibungen sind in der Projektdatei gespeichert.

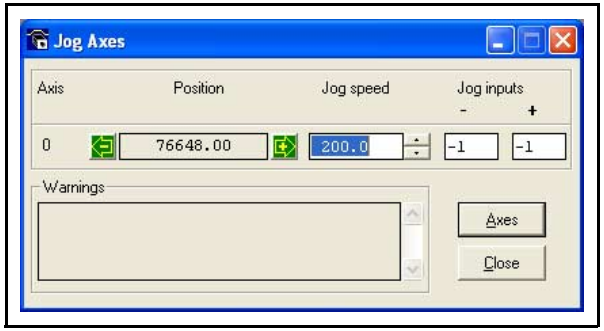
Tastenfeld

Für Trajexia nicht zutreffend.

Achsen-Tippbetrieb

Dieses Feld erlaubt es dem Benutzer, die Achsen im Trajexia-System zu bewegen.
In diesem Feld werden die bidirektionalen virtuellen E/A-Kanäle (16 bis 27) des Trajexia-Systems zum Einstellen der Jog-Eingänge verwendet. Die Vorwärts-, Rückwärts- und Schnelltippeingänge werden erkannt, indem zu den entsprechenden Achsenparametern geschrieben wird, und es wird davon ausgegangen, dass sie mit Öffner-Schaltern verbunden sind. Das bedeutet Folgendes: Wenn der Eingang eingeschaltet ist (+24 V liegen an), wird die entsprechende Jog-Funktion DISABLED (deaktiviert); wenn der Eingang ausgeschaltet ist (0 V), wird die Jog-Funktion ENABLED (aktiviert).
Die hier implementierten Jog-Funktionen deaktivieren die Schnelltippfunktion, d. h., dass die Drehzahl, bei der das Tippen durchgeführt wird, durch den Achsenparameter JOGSPEED vorgegeben wird. Darüber hinaus wird in diesem Feld die Tippgeschwindigkeit auf den Bereich 0. Solldrehzahl begrenzt, wobei die Solldrehzahl durch den Achsenparameter SPEED vorgegeben ist.
Bevor ein Tippvorgang initiiert wird, überprüft das Jog-Feld, ob alle im Jog-Feld und im Trajexia-System eingestellten Daten für die Durchführung eines Tippvorganges gültig sind.
Jog reverse (Rückwärts-Tippbetrieb): Mit dieser Schaltfläche wird ein Rückwärts-Tippvorgang initiiert. Um dies zu erreichen, wird die folgende Prüfsequenz durchgeführt

Abb.40



- Wenn es sich um eine SERVO-Achse handelt und der Servo AUS ist, wird eine Warnmeldung angezeigt.
- Wenn WatchDog AUS ist, wird eine Warnmeldung angezeigt.
- Wenn die Tippgeschwindigkeit 0 ist, wird eine Warnmeldung angezeigt.
- Wenn die Beschleunigungsrate für diese Achse 0 ist, wird eine Warnmeldung angezeigt.
- Wenn die Verzögerungsrate für diese Achse 0 ist, wird eine Warnmeldung angezeigt.
- Wenn der Rückwärtstippeingang außerhalb des Bereichs liegt, wird eine Warnmeldung angezeigt.
- Wenn bei diese Achse bereits eine Bewegung durchgeführt wird, die keine Tippbewegung ist, wird eine Warnmeldung angezeigt.

Jog forward (Vorwärts-Tippbetrieb): Mit dieser Schaltfläche wird ein Vorwärts-Tippvorgang initiiert. Es wird die gleiche Prüfung wie beim Rückwärts-Tippvorgang durchgeführt.
Wenn keine Warnmeldung ausgegeben wurde, wird die Meldung "**Forward jog set on axis?**" (Vorwärts-Tippvorgang für Achse einstellen?) im Warnungsfeld angezeigt; der **FAST_JOG**-Eingang für diese Achse wird ausgewertet, der Parameter **creep** (Schleichgang) wird auf den Wert eingestellt, der durch die Tippgeschwindigkeits-Steuerung vorgegeben ist und der Ausgang **JOG_FWD** wird ausgeschaltet, wodurch die Vorwärts-Tippfunktion aktiviert ist.

Jog speed (Tippgeschwindigkeit): Dies ist die Geschwindigkeit, mit der der Tippvorgang durchgeführt wird. In diesem Feld wird der Wert auf den Bereich von 0 bis zur gewünschten Drehzahl der diese Achse begrenzt, wobei die Solldrehzahl durch den Achsenparameter **SPEED** vorgegeben ist. Dieser Wert kann geändert werden, indem direkt zu diesem Steuerelement geschrieben oder die Tippgeschwindigkeits-Steuerung benutzt wird. Mit dem Rollbalken kann die Tippgeschwindigkeit nach oben oder nach unten in Schritten von 1 Einheit pro Sekunde geändert werden.

Jog inputs (Tipp-Eingänge): Dies sind die Eingänge, die mit der Vorwärts-/Rückwärts-Tippfunktion assoziiert sind. Sie müssen im Bereich von 8 bis zur Gesamtanzahl der Eingänge im System liegen, da die Eingangskanäle 0 bis 7 nicht bidirektional sind und daher der Zustand des Eingangs nicht durch den entsprechenden Ausgang eingestellt werden kann.

Der Eingang muss eingeschaltet sein, damit die Tippfunktion deaktiviert werden kann, und er muss ausgeschaltet sein, damit die Rückwärts-Tippfunktion aktiviert werden kann.
Wenn eine gültige Eingangsnummer eingestellt ist, wird deshalb der entsprechende Ausgang eingeschaltet und anschließend der entsprechende Achsenparameter **REV_JOG** gesetzt.
Axes (Achsen): Es wird ein Achsen-Auswahlfeld angezeigt, das es dem Benutzer ermöglicht, die Achsen auszuwählen, die im Display für die Achsenbewegung angezeigt werden sollen. Standardmäßig werden die an den Controller angeschlossenen physikalischen Achsen angezeigt.

Abb.41

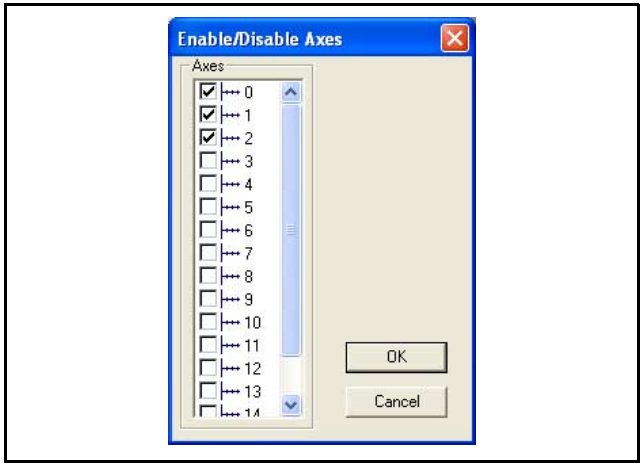
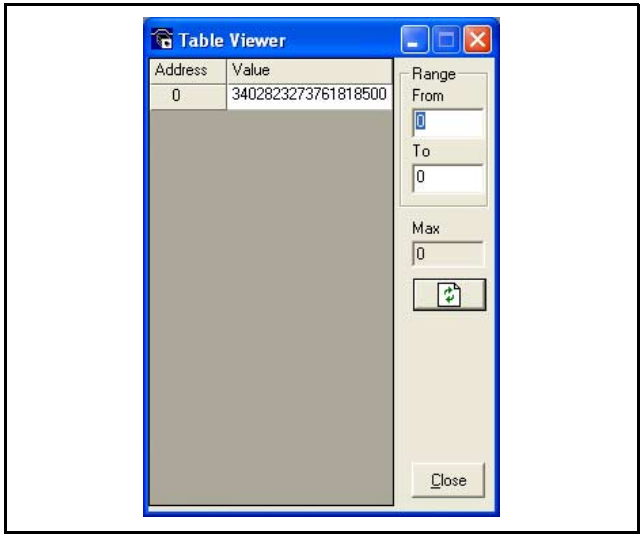


TABLE viewer (TABLE-Ansicht)

Die TABLE- und VR-Speicher-Editierwerkzeuge sind sehr ähnlich Eine Reihe von Werten im Speicher wird auf einer Oberfläche ähnlich einem Tabellenblatt angezeigt und kann modifiziert werden. Um einen Wert zu modifizieren, klicken Sie auf einen existierenden Wert, geben Sie den neuen Wert ein, und drücken Sie die Eingabetaste. Die Änderung wird sofort aktiv und kann auch bei laufenden Programmen vorgenommen werden.
Options (Optionen):

- **Range (Bereich)**
  Beide Tools haben die Option, den Anfang und das Ende des Anzeigebereiches einzustellen. Im TABLE-Ansichtswerkzeug stellt der Maximalwert den höchsten lesbaren Wert dar (dies ist der Systemparameter **TSIZE**).
  Wenn der Wertebereich größer ist als im Dialogfeld angezeigt werden kann, wird für die Liste eine Bildlaufleiste angezeigt, damit alle Werte betrachtet werden können.

Abb.42



- Schaltfläche „Refresh“ (Aktualisieren)
  Dieser Bildschirm wird nicht automatisch aktualisiert; wenn ein TABLE- oder VR-Wert vom Programm geändert wird, können Sie den neuen Wert nicht sehen, bis die Anzeige aktualisiert wird.

Watch variables (Variablen beobachten)

In Trajexia nicht implementiert.

Analog inputs (Analogeingänge)

Überwacht den Wert im dezentralen analogen Eingangsmodul. Die Eingänge werden automatisch zum System hinzugefügt, beginnend bei AIN0, wenn eines oder mehrere AN2900-Module erkannt werden.

Terminal

Das Terminalfenster ist ein Texteditor, der eine direkte Verbindung zum Trajexia-System hat. Die meisten Funktionen, die während der Installation, Programmierung und Inbetriebnahme eines mit Trajexia ausgestatteten Systems durchgeführt werden müssen, sind durch die Optionen automatisiert worden, die in den Menüoptionen von Trajexia Tools verfügbar sind. Wenn jedoch eine direkte Kommunikation erforderlich ist, kann das Terminalfenster verwendet werden.

Abb.43

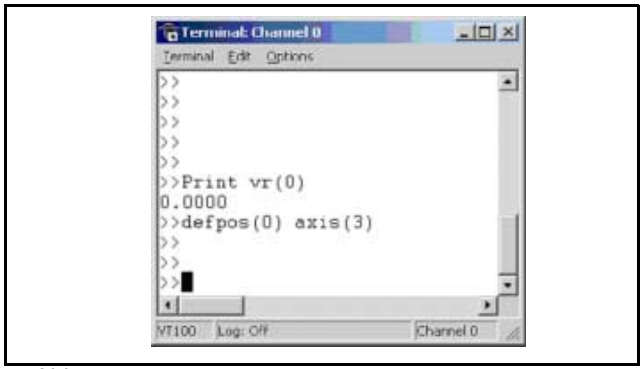
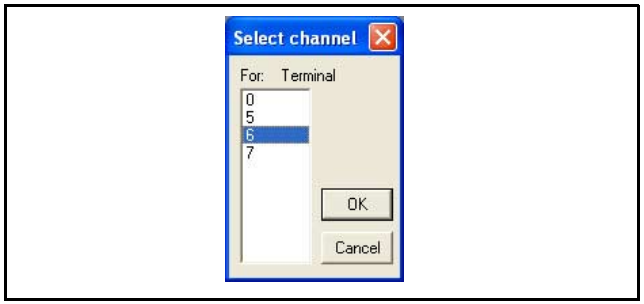


Abb.44



Kanal auswählen. Wenn Trajexia Tools mit dem Controller verbunden ist, zeigt das Terminal-Tool ein Dialogfeld zur Auswahl des Kommunikationskanals.

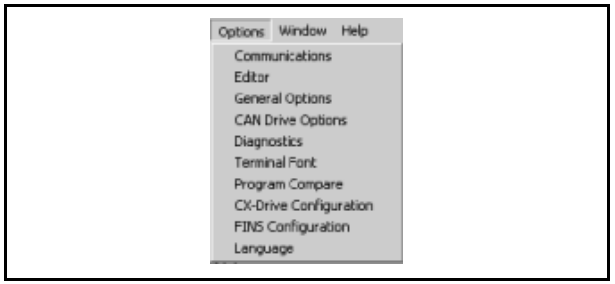
Kanal 0 wird für die Trajexia-Befehlszeile benutzt, und die Kanäle 5, 6 und 7 dienen zur Kommunikation mit Programmen, die unter Trajexia laufen.

Wählen Sie den benötigten Kanal, und klicken Sie auf **OK**, um ein Terminal-Tool für den ausgewählten Kanal zu starten. Nur ein Terminal-Tool (oder Tastenfeld-Tool) kann zu einer Zeit mit einem Kanal verbunden werden.

5.5.5 Das Menü „Options“ (Optionen)

Im Menü **Options** werden die Systemoptionen für das Trajexia-System eingestellt:

Abb.45

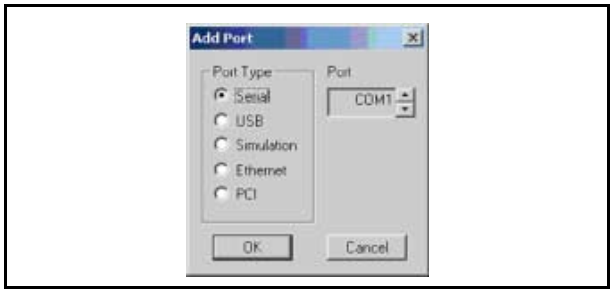


Communications (Kommunikation)

Ermöglicht die Einstellung und Betrachtung der Kommunikationseinstellungen. Die Einstellungen können nur offline geändert werden. Die verschiedenen Optionen sind:

- Seriell für andere OMRON Motion-Controller (C200HW-MC402-E und R88-MCW151-E).
- USB wird nicht verwendet.
- Simulation. Um offline arbeiten zu können, wird ein virtuelles Motion-Control-System simuliert.
- Ethernet ist die für Trajexia verwendete Option.
- PCI wird nicht verwendet.

Abb.46



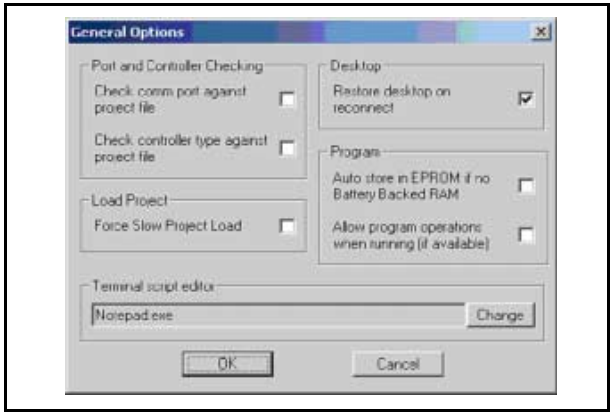
Editor

Dient zur Bearbeitung der verschiedenen Optionen des Texteditors.

General Options (Allgemeine Optionen)

Ermöglicht die Einstellung verschiedener Optionen des Systems.

Abb.47

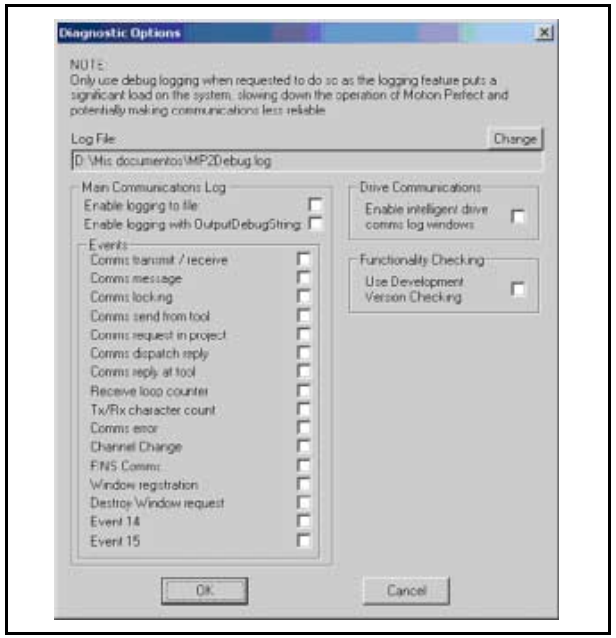


CAN Drive (CAN-Treiber)

In den Optionen nicht implementiert.

Diagnostics (Diagnose)
Ermöglicht die Auswahl von Ereignissen, um sie in einer **.log**-Datei zu Diagnosezwecken zu speichern.

Abb.48



Terminal Font (Terminal-Schriftart)
Dient zur Auswahl der Schriftart, die im Terminalfenster angezeigt werden soll. Sehr nützlich für die Inbetriebnahme.

Program Compare (Programmvergleich)
Ermöglicht das Vergleichen von Programmen.

CX-Drive Configuration (CX-Drive-Konfiguration)
Ermöglicht die Auswahl des Verzeichnisses der CX-Drive-Datenbank.

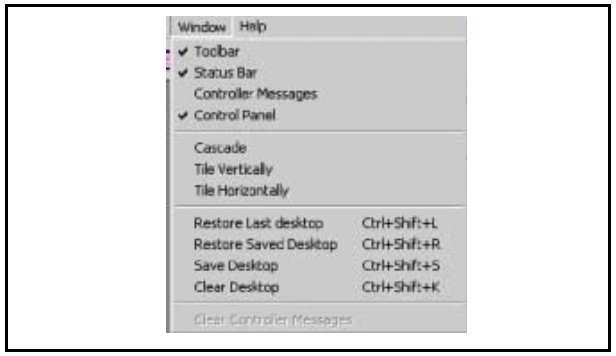
FINS Configuration (FINS-Konfiguration)
Ermöglicht die Auswahl der Schnittstelle und der Zeitüberschreitung für die FINS-Kommunikation.

Language (Sprache)
Zur Zeit ist nur Englisch implementiert.

5.5.6 Das Menü „Windows“ (Anzeige)

- Restore Last desktop/Restore Saved Desktop/Save Desktop/ Clear Desktop (Letzten Desktop wiederherstellen/Gespeicherten Desktop wiederherstellen/Desktop speichern/Desktop löschen): Dies sind Tools zum schnellen Verwalten und Konfigurieren Ihres Desktops nach den Bedürfnissen des Benutzers.
- Clear Controller Messages (Controller-Meldungen löschen): Löscht das Fenster **Controller Messages**.

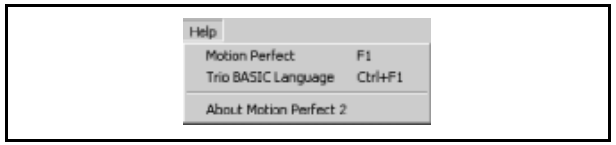
Abb.49



5.5.7 Das Menü „Help“ (Hilfe)

- Motion Perfect: Die Hilfe von Trajexia Tools.
- Trio BASIC Language (Trio BASIC-Sprache): Hilfe für die BASIC-Befehle und -parameter.
- About Motion Perfect 2 (Über Motion Perfect 2): Zeigt die Version von Trajexia Tools an.

Abb.50



6 Beispiele und Tipps

In diesem Kapitel werden zwei Kategorien von Beispielen und Tipps gegeben:
- Anleitungen.
- Praktische Beispiele

6.1 Anleitungen

6.1.1 STARTUP-Programm

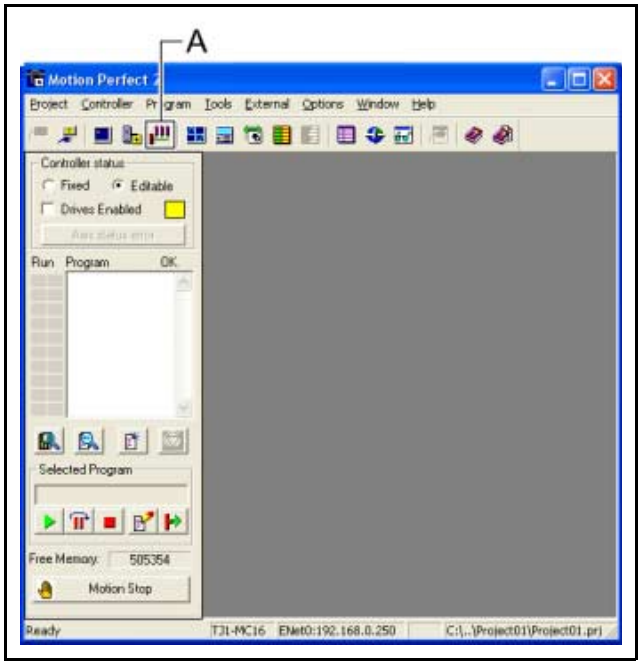
Der Zweck dieses Programms besteht darin, die erkannte MECHATROLINK-II-Konfiguration mit der erwarteten zu vergleichen (die erwartete Konfiguration ist die Konfiguration, die in dem Moment besteht, in dem Sie das Programm erstellen).
Das STARTUP-Programm führt folgende Maßnahmen durch:
- Die Anzahl der Teilnehmer im System wird überprüft.
- Es wird überprüft, ob die Teilnehmernummern übereinstimmen.
- Es wird überprüft, ob alle Geräte angeschlossen und mit Spannung versorgt sind.
- Liegt eine Abweichung vor, stoppt das Programm.
- Der richtige Parameter für **ATYPE** wird eingestellt, wie er im Fenster für die intelligente Achse ausgewählt wurde. Die Betriebsart wird eingestellt, **Run** (Betrieb) oder **Commisioning** (Inbetriebnahme).

Verwendung des STARTUP-Programms

Die empfohlene Methode zur Verwendung des **STARTUP** -Programms ist folgende:

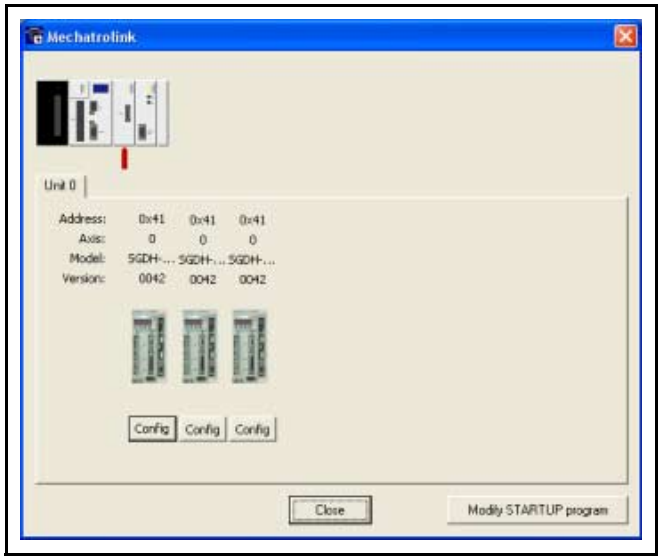
1. Klicken Sie auf die Schaltfläche **Intelligent drives** (Intelligente Antriebe) (A).

Abb.1



2. Klicken Sie auf die Schaltfläche **Modify STARTUP program** (STARTUP-Programm modifizieren).
3. Geben Sie nach dem Ende der automatisch erstellten Sektion Ihren eigenen Anwendungscode ein. Typischerweise Variablen-Initialisierung und Achsenparameter.
4. Wenn das STARTUP-Programm ausgeführt ist, starten Sie Ihre Anwendungsprogramme. Es wird empfohlen, das Programm „SHELL“ zu starten, das später in diesem Abschnitt erklärt wird.
5. Stellen Sie ein, dass das STARTUP-Programm beim Einschalten der Spannungsversorgung ausgeführt werden soll.

Abb.2



Hinweis
OMRON empfiehlt, am Ende des STARTUP-Programms die Anweisung **RUN „ihr_programm”** einzugeben, um Ihr Anwendungsprogramm zu starten. Das Anwendungsprogramm startet, wenn das START-Programm erfolgreich und ohne Fehler ausgeführt ist.
Wenn Sie Ihr Anwendungsprogramm für „Run at startup” (beim Hochfahren ausführen) einstellen, besteht das Risiko, dass die Maschine startet, wenn ein Fehler im MECHATROLINK-II-Bus vorliegt.

Beispiel

```
'================================================
'Der Erste Teil des Programms wird automatisch
'durch das Fenster fUEr intelligente Achsen in TRAJEXIA TOOLS
'GENERIERT. Es besteht aus einer PrUEfsequenz, um
festzustellen,
'ob es sich bei der erkannten Achsenkonfiguration um
'die erwartete handelt.
'Wenn ja, wird das Programm beendet und "SHELL" gestartet.
'Wenn nicht, wird das Programm gestoppt und kein
anderes Programm gestartet.
'Dieses Programm muss so eingestellt werden, dass es
beim Einschalten der Spannungsversorgung in 'einer
'Task mit niedriger PrioritAEt (in diesem Beispiel 1)
ausgefUEhrt wird.
'================================================
'MECHATROLINK-Abschnitt starten
' Erkannte Geraete überpruefen
' Baugruppe 0
IF NOT MECHATROLINK(0,3,0) THEN
    PRINT "Fehler bei der Geraetezaehlung fuer Baugruppe 0"
    STOP
ELSE
    IF VR(0) <> 3 THEN
        PRINT "Falsche Geraetezaehlung fuer Baugruppe 0"
        STOP
    ENDIF
ENDIF
```

```
IF NOT MECHATROLINK(0,4,0,0) THEN
    PRINT "Fehler bei der Adressenerfassung für
    Baugruppe 0, Station 0"
    STOP
ELSE
    IF VR(0) <> 65 THEN
       PRINT "Falsche Adresse für Baugruppe 0, Station 0"
        STOP
    ENDIF
ENDIF
IF NOT MECHATROLINK(0,4,1,0) THEN
    PRINT "Fehler bei der Adressenerfassung für
    Baugruppe 0, Station 1"
    STOP
ELSE
    IF VR(0) <> 66 THEN
       PRINT "Falsche Adresse fuer Baugruppe 0, Station 1"
        STOP
    ENDIF
ENDIF
IF NOT MECHATROLINK(0,4,2,0) THEN
    PRINT "Fehler bei der Adressenerfassung für
    Baugruppe 0, Station 2"
    STOP
ELSE
    IF VR(0) <> 67 THEN
       PRINT "Falsche Adresse fuer Baugruppe 0, Station 2"
        STOP
    ENDIF
ENDIF
' Achsentypen einstellen
' Baugruppe 0
ATYPE AXIS(0)=40
ATYPE AXIS(1)=40
ATYPE AXIS(2)=40
' Antriebe in RUN-Betriebsart setzen
' Baugruppe 0
MECHATROLINK(0,20,65)
MECHATROLINK(0,20,66)
MECHATROLINK(0,20,67)
'MECHATROLINK-Abschnitt stoppen
'================================================
```

```
'dieser Abschnitt muss manuell vom Benutzer
entsprechend der Anwendung
'eingestellt werden. Typische MaSSnahmen sind
'VARIABLENINITIALISIERUNG, SERVO-/ ACHSENEINSTELLUNG,
BENENNUNG
'GLOBALER VARIABLEN UND STARTEN DES "SHELL"-PROGRAMMS.
'================================================
'Namen für globale Variablen definieren
GLOBAL "projekt_status",100
GLOBAL "alarm_status",101
GLOBAL "aktion",102
'Variablen initialisieren
VR(0)=0
projekt_status=0
alarm_status=0
aktion=0
'SHELL-Programm starten
RUN "SHELL",2
STOP
```

6.1.2 Verstärkungseinstellungen

Die Verstärkungseinstellung richtet sich nach dem mechanischen System, an das der Motor angeschlossen ist. Es gibt drei Hauptkonzepte:

- Trägheitsverhältnis
- Steifigkeit
- Resonanzfrequenz.

Diese Konzepte sind im Hardware-Referenzhandbuch im Kapitel Systemphilosophie beschrieben.

Dieser Abschnitt zeigt Beispiel-Parameterwerte für:

- Drehzahlregelkreisverstärkung
- Proportionale Positionsverstärkung
- Verstärkung der Drehzahlvorsteuerung.

Die Beispielwerte für das Programm und die Bewegungsparameter im Trajexia-System sind unten angegeben. Bitte beachten Sie, dass diese Werte für 13-Bit-Drehgeber geeignet sind.

Antriebs-Parameterwert	Beschreibung
Pn103 = 716	Trägheitsverhältnis
Pn110 = 0012	Kein Autotuning
Pn202=1	Übersetzungsverhältnis-Zähler
Pn203=1	Übersetzungsverhältnis-Nenner

Bewegungs-Parameterwerte	Beschreibung
UNITS =1	Angabe in Drehgeberimpulsen
SPEED=200000	Drehzahleinstellung
ACCEL=1000000	Beschleunigungseinstellung
DECEL=1000000	Verzögerungseinstellung
MOVEMENT=81920	10 Umdrehungen

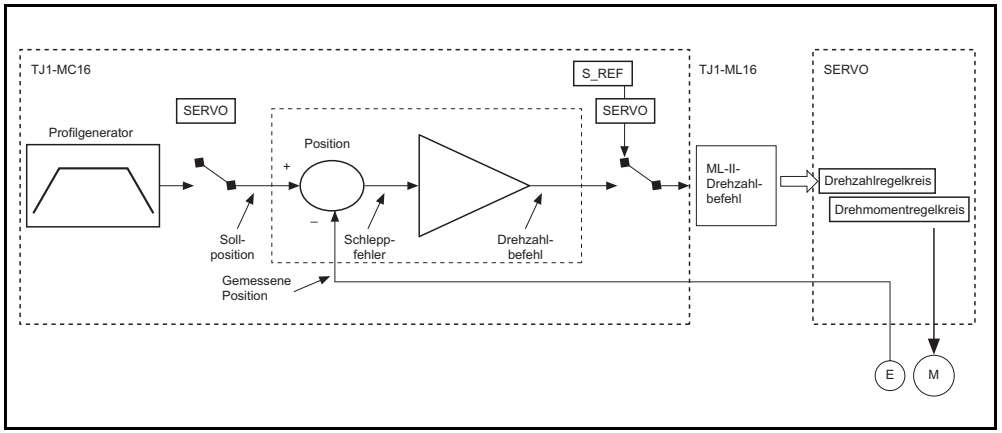
Beispiele für Geschwindigkeits-Betriebsart Modus

In dieser Betriebsart wird der Positionsregelkreis in Trajexia und der Drehzahlregelkreis im Servotreiber geschlossen. Der Achsenparameter **Speed** (Drehzahl) wird über das MECHATROLINK-II-Netzwerk zum Servotreiber gesendet und liest die Positionsrückmeldung.

```
BASE(0)
ATYPE=41 'MECHATROLINK Geschwindigkeits-Betriebsart
SERVO=1
WDOG=1
DEFPOS(0)
schleife:
    MOVE(81920)
    WAIT IDLE
    WA(100)
    DEFPOS(0)
GOTO schleife
```

Beispiel 1

Abb.3

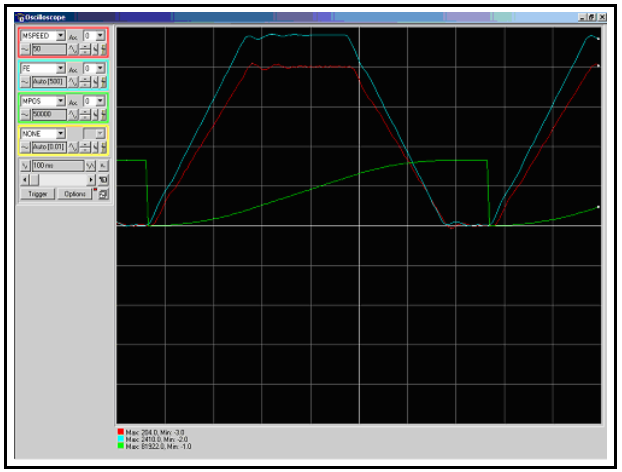


Nur die Proportionalverstärkung hat einen festen Wert,
der Schleppfehler ist proportional zur Drehzahl.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=0
Fn001=4

Hinweis:
Die Farben und die Skala des Oszilloskops für die Geschwindigkeits-Betriebsart sind wie folgt:
Rot: MSPEED (gemessene Achsendrehzahl).
Die Einheiten werden in 50 Einheiten/ms/Unterteilung gemessen
Blau: FE (Schleppfehler). Die Einheiten hängen vom Graph ab.
Grün: MPOS (gemessene Achsenposition).
50000 Einheiten/Unterteilung

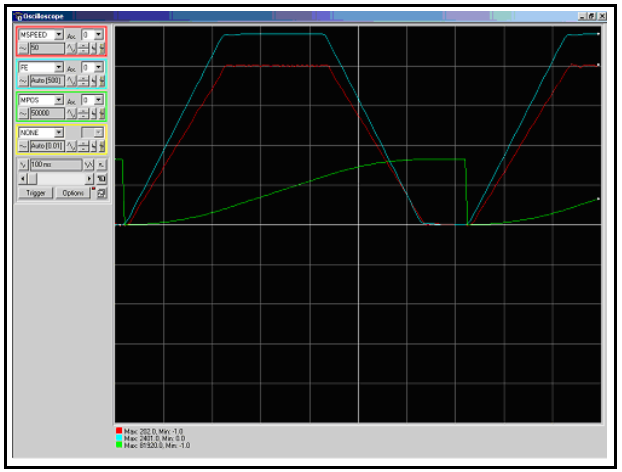
Abb.4



Beispiel 2
Der Wert für die Steifigkeit wird erhöht. Die Fehlergröße bleibt gleich, aber die Welligkeit, die Drehzahlstabilität und das Überschwingen sind besser.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=0
Fn001=6

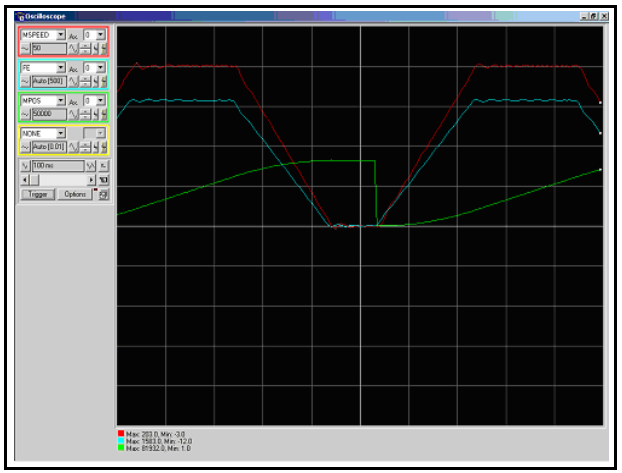
Abb.5



Beispiel 3
Der Parameter **P_GAIN** wird weiter erhöht.
Der Schleppfehler verringert sich proportional.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =200000
VFF_GAIN=0
Fn001=6

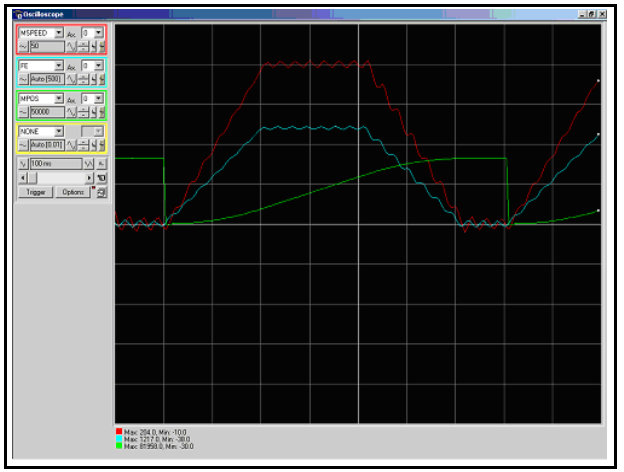
Abb.6



Beispiel 4
Der Wert des Parameters **P_GAIN** ist zweimal so hoch wie der Wert in Beispiel 1. Der Schleppfehler beträgt die Hälfte, aber auf Grund der übermäßigen Verstärkung treten Vibrationen auf.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =262144
VFF_GAIN=0
Fn001=6

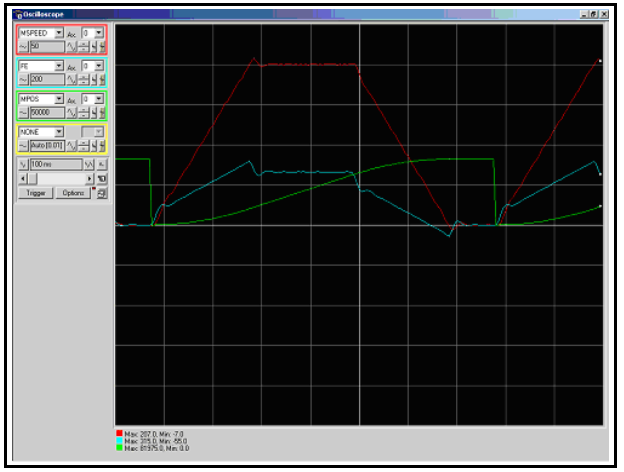
Abb.7



Beispiel 5
Der Wert des Parameters **P_GAIN** wird auf den Wert aus Beispiel 1 gesetzt. Der Wert von **VFF_GAIN** wird erhöht. Der Schleppfehler wird ohne eine Verringerung der Stabilität reduziert. Der Schleppfehler ist nicht proportional zur Drehzahl.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=1400000
Fn001=6

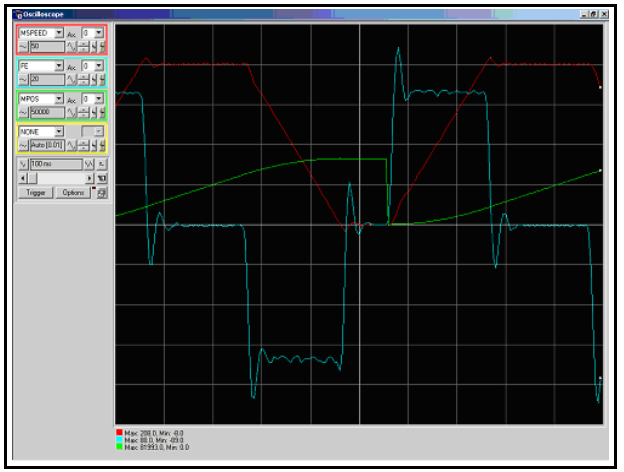
Abb.8



Beispiel 6
Mit diesem Wert von VFF_GAIN ist der Schleppfehler proportional zur Beschleunigung und kleiner als bei nur proportionaler Verstärkung (die Skalierung beträgt 20 Einheiten/Unterteilung). Bei konstanter Drehzahl nähert sich der Schleppfehler null.
Die negativen Auswirkungen dieser Werte sind Überschwingen und Unterschwingen, wenn sich die Beschleunigung ändert; dies kann durch Erhöhung der Drehzahlregelkreisverstärkung verringert, aber nicht vermieden werden, wenn das mechanische System eine hohe Verstärkung verkraftet kann.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=1573500
Fn001=6

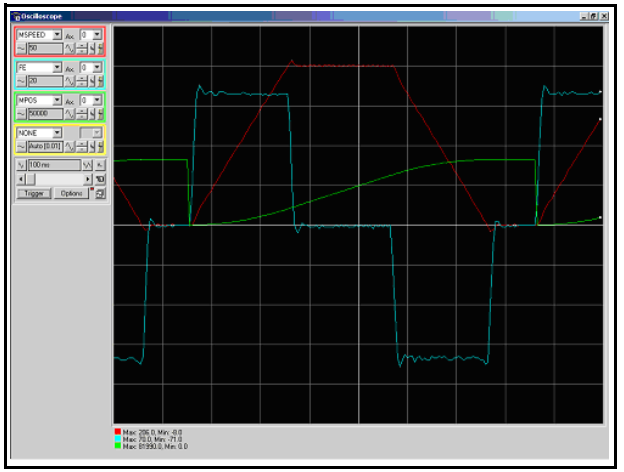
Abb.9



Beispiel 7
Der Wert der Steifigkeit wird von 6 auf 8 erhöht. Das Über-/Unterschwingen ist geringer, aber die Vibrationen des Motors sind stärker.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=1573500
Fn001=8

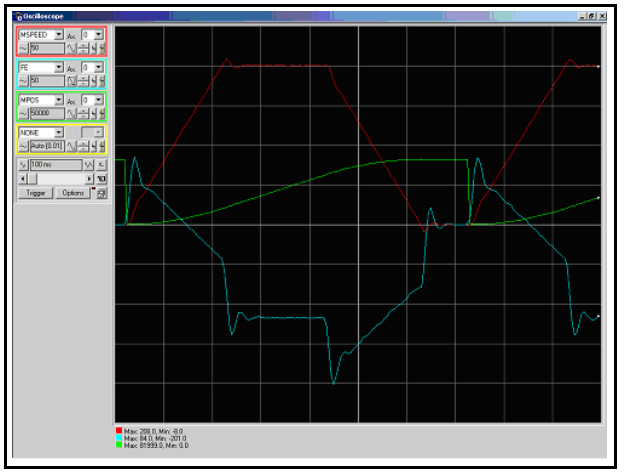
Abb.10



Beispiel 8
Im Gegensatz zu P_GAIN, bei dem die Regel „je höher, desto besser“ gilt (der Grenzwert ist erreicht, wenn das mechanische System zu vibrieren anfängt), gibt es für den Parameter VFF_GAIN einen Optimalwert (den aus Test 6); bei Werten, die höher als dieser Wert sind, tritt ein Fehler auf, der proportional zur Drehzahl/Beschleunigung ist, aber mit anderem Vorzeichen. Die erforderliche Korrektur ist zu groß.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
P_GAIN =131072
VFF_GAIN=1650000
Fn001=6

Abb.11

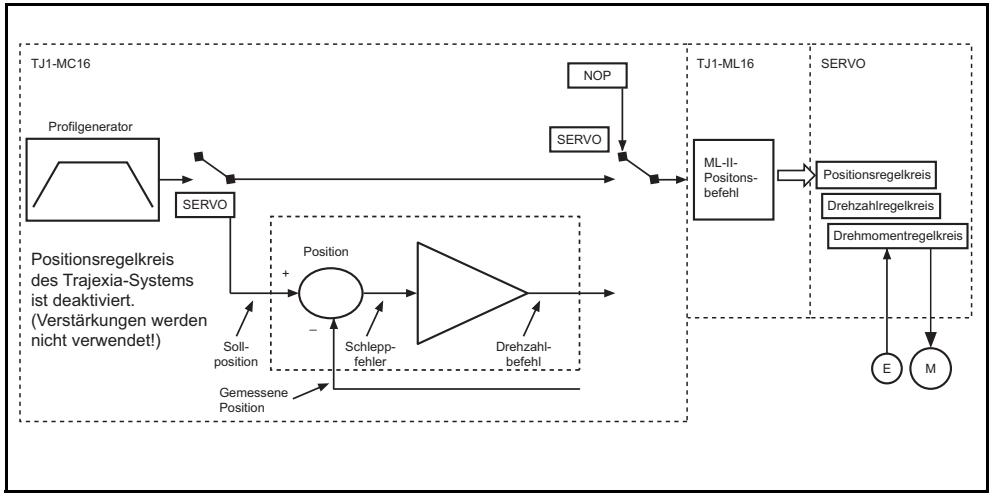


Beispiele für Positions-Betriebsart

In dieser Betriebsart sind der Positions- und Drehzahlregelkreis im Servotreiber geschlossen. Der TJ1-ML__ sendet den Positionsbefehl über das MECHATROLINK-II-Netzwerk zum Servotreiber und liest die Positionsrückmeldung.
Bitte beachten Sie, dass dieses System keine Abtastverzögerung im Vergleich zum Positionsregelkreis im Servotreiber hat, der Soll_position in Zyklus „n“ mit der Gemessenen_Position in Zyklus „n“.
Für den internen Gebrauch verwendet Trajexia weiterhin seinen eigenen Positionsregelkreis, deshalb ist der Schleppfehler, der im Achsenparameter in Trajexia gelesen wird, nicht der echte im Servoantrieb. Um den richtigen Schleppfehler auszulesen, verwenden Sie DRIVE_MONITOR.
Passen Sie die Steifigkeit des Servos, die Drehzahlregelkreisverstärkung und die Positionsverstärkung gleichzeitig an, indem Sie nur die proportionale Positionsverstärkung verwenden.
Die Ergebnisse sind ähnlich wie bei der MECHATROLINK-II-Geschwindigkeits-Betriebsart, mit folgenden Vorteilen:

- Die Einstellung ist einfacher, nur die Steifigkeit (Fn001) und – falls nötig – die Verstärkung der Vorsteuerung (Pn109) müssen eingestellt werden.

Abb.12



- Der Positionsregelkreis im Servo ist schneller (250 µs) als in Trajexia und wird zusammen mit dem Drehzahlregelkreis durchlaufen.
- Es gibt keine Abtastzeitverzögerung zwischen „Zielposition“ und „Gemessener Position“.

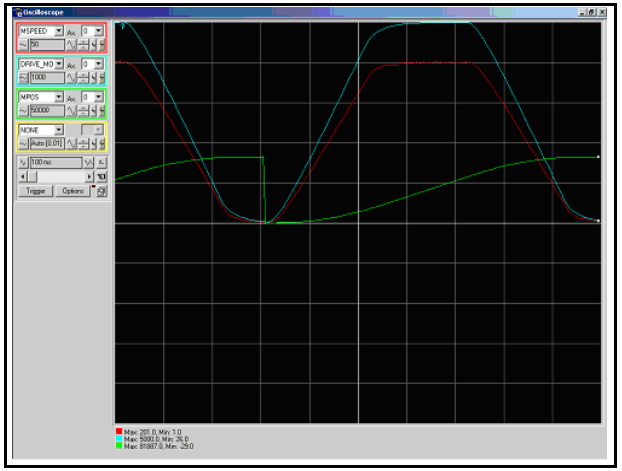
Zur Feinabstimmung können die verschiedenen Verstärkungsparameter einzeln geändert werden.

```
BASE(0)
ATYPE=41 'MECHATROLINK-Positions-Betriebsart
SERVO=1
DRIVE_CONTROL=2 'um den Schleppfehler zu überwachen in
                'DRIVE_MONITOR
WDOG=1
DEFPOS(0)
Regelkreis:
    MOVE(81920)
    WAIT IDLE
    WA(100)
    DEFPOS(0)
GOTO schleife
```

Beispiel 1

Der Schleppfehler ist proportional zur Drehzahl. Es ergibt sich ein „weiches Profil“ auf Grund der niedrigen Einstellung für die Steifigkeit (geringe Verstärkung).

Abb.13



Hinweis:
Die Farben und die Skala des Oszilloskops für die Positions-Betriebsart sind wie folgt:
Rot: MSPEED (gemessene Achsendrehzahl).
Die Einheiten werden in 50 Einheiten/ms/Unterteilung gemessen.
Blau: DRIVE_MONITOR (als Schleppfehler im Servotreiber eingestellt). Die Einheiten hängen vom Graph ab.
Grün: MPOS (gemessene Achsenposition).
50000 Einheiten/Unterteilung

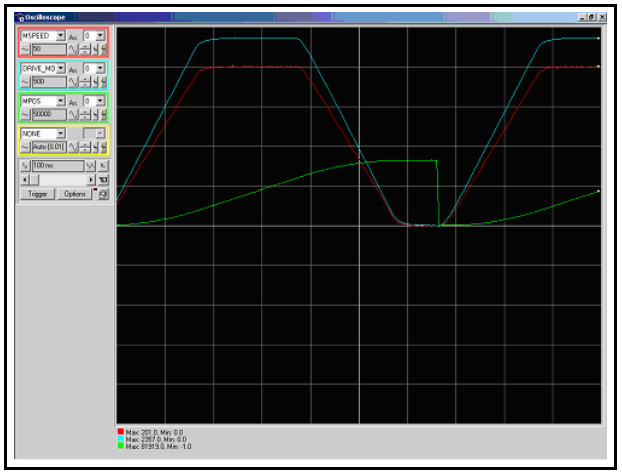
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
Fn001=4
Pn109=0

Beispiel 2
Der Schleppfehler verringert sich mit steigender Steifigkeit.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
Fn001=6
Pn109=0

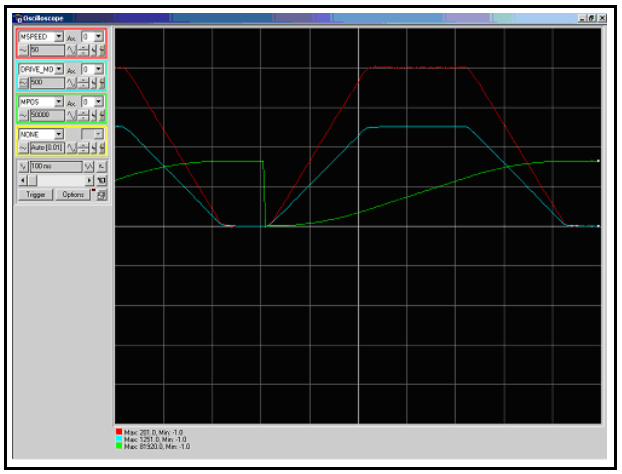
Abb.14



Beispiel 3
Bei hoher Verstärkung fängt der Motor an zu vibrieren, aber das Profil ist stabiler als in der MECHATROLINK-II-Geschwindigkeit-Betriebsart.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
Fn001=8
Pn109=0

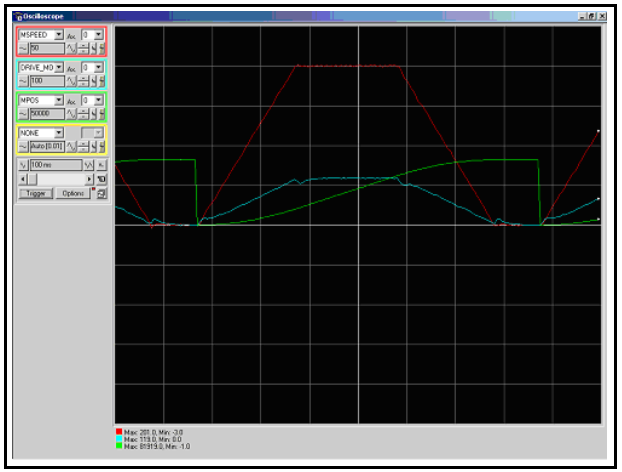
Abb.15



Beispiel 4
Die Auswirkung der Verstärkung der Drehzahlvorsteuerung besteht darin, dass der Schleppfehler verringert wird und die Auswirkung proportional zur Beschleunigung ist.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
Fn001=6
Pn109=95

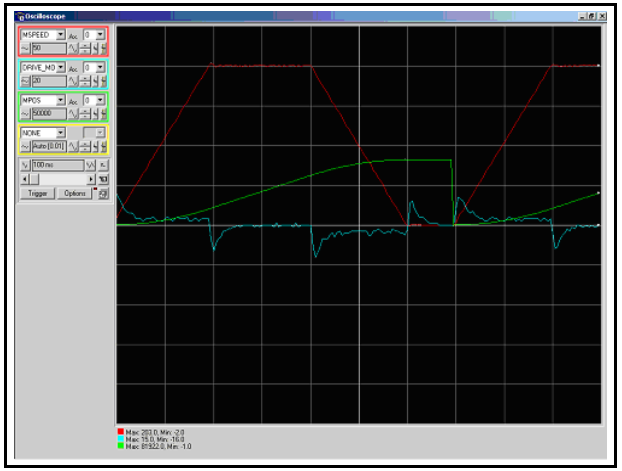
Abb.16



Beispiel 5
Wird die Verstärkung der Drehzahlvorsteuerung auf 100 % gesetzt, ist der Schleppfehler sehr gering und proportional zur Beschleunigung. Der Optimalwert von 100 % Drehzahlvorsteuerung ist der Maximalwert, der eingestellt werden kann. Der Parameterwert von Pn109 ist einfacher einzustellen als der Parameterwert von VFF_GAIN.
Die Parameterwerte für das Beispiel sind:

Bewegungs-Parameterwerte
Fn001=6
Pn109=100

Abb.17



6.1.3 Einstellung des Achsenparameters UNITS und des Übersetzungsverhältnisses

Wenn die mechanische Achse mit dem Trajexia TJ1-MC__, einem Servotreiber und einem Servomotor gesteuert wird, sind Drehgeberimpulse die einzigen Maßeinheiten, die die Hardware versteht. Alle Befehle an den Treiber zur Bewegung einer Achse werden in Drehgeberimpulsen ausgedrückt. Alle Rückmelde-Informationen über Achsenpositionen werden ebenfalls in Drehgeberimpulsen ausgedrückt. Wenn Programme in BASIC geschrieben werden, um Bewegungen oder eine Folge von Bewegungen zu erreichen, wird es der Benutzer vorziehen, mit benutzerdefinierten Einheiten zu arbeiten, z. B. Millimeter, Zentimeter, Meter, Grad, „Produkt“, „Drehung“, „Stationen“.
Der Achsenparameter **UNITS** enthält den Umwandlungsfaktor zwischen Drehgeberimpulsen und benutzerdefinierten Einheiten. Alle Achsenparameter, die mit Bewegungen und Argumenten von Achsenbefehlen in Verbindung stehen und den Grad einer Bewegung bestimmen, werden in diesen Benutzereinheiten ausgedrückt. Dieser Parameter ermöglicht es dem Benutzer, die Einheiten zu definieren, mit denen es sich am angenehmsten arbeiten lässt. Beispielsweise können Sie für ein bewegliches Teil, das eine lineare Bewegung beschreibt, die Einheit mm oder mm-Bruchteile festlegen. Bei einem beweglichen Teil, das eine Drehbewegung beschreibt, können Grad oder Grad-Bruchteile verwendet werden. Weitere Informationen zum Achsenparameter **UNITS** finden Sie in Abschnitt 3.2.270.
Bitte beachten Sie, dass die Umwandlung von Drehgeberimpulsen zu benutzerdefinierten Einheiten nicht nur mit dem Achsenparameter **UNITS** möglich ist. Bestimmte Servotreiber-Parameter und einige Merkmale des mechanischen Systems sind ebenfalls wichtig. In den folgenden Abschnitten wird beschrieben, welche Servotreiber-Parameter für diese Umwandlung wichtig sind. Außerdem geben wir Beispiele dafür, wie dieser Parameter und der Achsenparameter **UNITS** eingestellt werden, wobei die Merkmale des mechanischen Systems berücksichtigt werden.

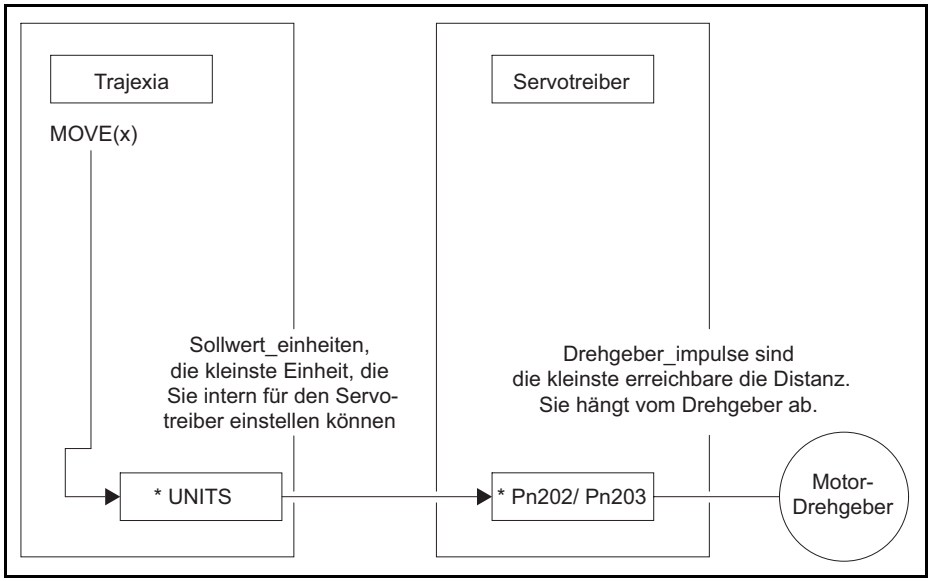
Umwandlung von Drehgeberimpulsen zu benutzerdefinierten Einheiten

Zwei sehr wichtige Parameter des Sigma-II-Servotreibers für die Umwandlung von Drehgeberimpulsen in Benutzereinheiten sind Pn202 und Pn203. Wenn ein Servomotor mit einem Absolutwert-Drehgeber verwendet wird, ist es außerdem notwendig, den Parameter Pn205 einzustellen.
Der Parameter Pn202 ist der Nenner für das elektronische Übersetzungsverhältnis (G1). Der Parameter Pn203 ist der Zähler für das elektronische Übersetzungsverhältnis (G2).
Der Servomotor dreht unter Verwendung des Wertes des vom TJ1-MC__ gesendeten Positionsbefehlssignals, multipliziert mit dem elektronischen Übersetzungsverhältnis (Pn202, Pn203). Auf der Ausgangs- (Servomotor-) seite wird das Signal als Anzahl von Drehgeberimpulsen ausgedrückt. Weitere Informationen zu den Servotreiber-Parametern Pn202 und Pn203 finden Sie im Sigma-II-Servotreiber-Handbuch.
Der Achsenparameter UNITS drückt effektiv das Verhältnis zwischen den Benutzereinheiten aus, die der Benutzer im Programm verwenden will und der Position, die über den MECHATROLINK-II-Bus an den Servotreiber gesendet wird. Unter Berücksichtigung des elektronischen Übersetzungsverhältnisses ist die Gleichung, die das Verhältnis zwischen Benutzereinheiten, dem Parameter **UNITS**, den Parametern Pn202 und Pn203, den Drehgeberimpulsen und den mechanischen Maßeinheiten wiedergibt, wie folgt:

$$\frac{Pn202}{Pn203} \cdot UNITS = \frac{y \cdot Drehgeber_impulse}{x \cdot Benutzer_einheiten}$$

Hierbei ist *y* die Anzahl der Drehgeberimpulse und *x* der Betrag in Benutzereinheiten.

Abb.18



Beispiel 1

Das mechanische System besteht aus einem einfachen Drehtisch. Es wird ein Servomotor mit 13-Bit-Inkrementaldrehgeber verwendet. Das Übersetzungsverhältnis des Getriebes ist 1:10. Die gewünschte Benutzereinheit ist Grad. Das System kann mit den folgenden Gleichungen beschrieben werden:

$$1 \cdot Motor_umdrehung = 2^{13} \cdot Drehgeber_impulse$$

$$10 \cdot Motor_umdrehung = 1 \cdot Maschinen_zyklus$$

$$1 \cdot Maschinen_zyklus = 360^{\circ}$$

Die Kombination dieser Gleichungen hat folgendes Ergebnis:

$$\frac{Pn202}{Pn203} \cdot UNITS = \frac{2^{13} \cdot Drehgeber_impulse}{1 \cdot Motor_umdrehung} \frac{10 \cdot Motor_umdrehungen}{1 \cdot Maschinen_umdrehung} \frac{1 \cdot Maschinen_umdrehung}{360^{\circ}} =$$

$$\frac{2^{13} \cdot 10}{360} \frac{Drehgeber_impulse}{Grad}$$

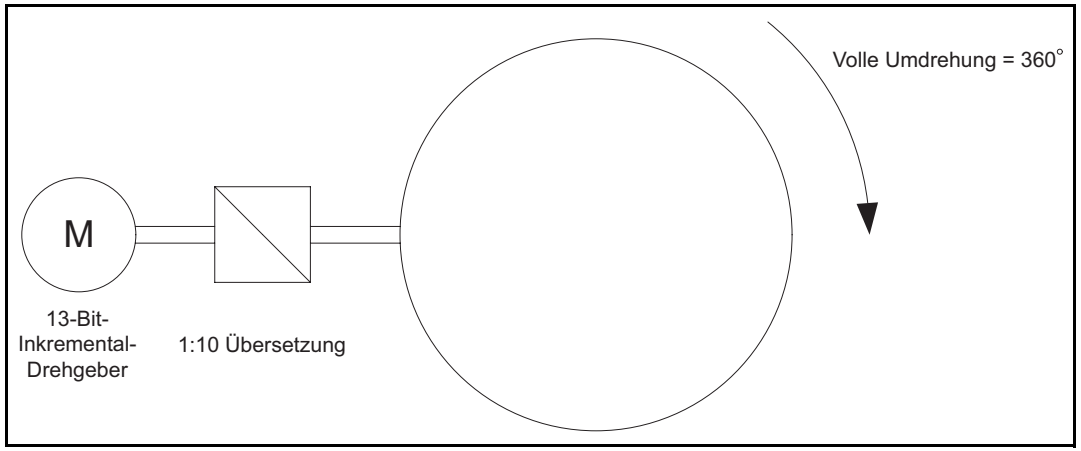
Deshalb gilt:

$$\frac{Pn202}{Pn203} \cdot UNITS = \frac{2^{13} \cdot 10}{360}$$

Aus dieser Gleichungen können wir die Werte für Pn202, Pn203 und **UNITS** ableiten, wobei sich folgende Einschränkungen und Empfehlungen ergeben:

1. Pn202 und Pn203 sind Ganzzahlen.
2. UNITS darf keine unbegrenzte Anzahl an Dezimalstellen haben. Dadurch können Rundungsfehler entstehen, die kleine Positionsfehler zur Folge haben, die sich zu großen Gesamt-Positionsfehlern summieren.
3. Aus Stabilitätsgründen ist es notwendig, Einstellungen zu vermeiden, in denen Pn202/Pn203 kleiner als 0,01 oder größer als 100 ist.

Abb.19



Wir können nun die letzte Gleichung folgendermaßen umschreiben:

$$UNITS \cdot \frac{Pn202}{Pn203} = 2^{13} \frac{10}{360}$$

Eine Lösung für diese Gleichung ist:

$$UNITS = 2^{13} = 8192$$
$$Pn202 = 10$$
$$Pn203 = 360$$

Wenn wir die dritte Empfehlung aus der obigen Liste berücksichtigen (Einstellungen zu vermeiden, in denen Pn202/Pn203 kleiner als 0,01 oder größer als 100 sind), können wir die letzte Gleichung wie folgt umschreiben:

$$UNITS \cdot \frac{Pn202}{Pn203} = 2^{13} \frac{10}{360} = 2^{8} \frac{2^{5}}{36} = 2^{8} \frac{32}{36}$$

Dadurch ergibt sich folgende Lösung:

$$UNITS = 2^{8} = 256$$
$$Pn202 = 32$$
$$Pn203 = 36$$

Mit diesen Werten wird der Tisch durch den Befehl **MOVE(28)** 28 Grad in positive Richtung gedreht.

Absolutwert-Drehgebereinstellung

Der Absolutwert-Drehgeber hält die aktuelle Motorposition, auch wenn keine Spannung anliegt. Der Absolutwert-Drehgeber gibt die Position innerhalb einer Umdrehung vor (d.h. einen Bruchwert zwischen 0 und 1 außer 1) und hat einen Multi-Umdrehungszähler. Sie können das Umdrehungsverhalten des Absolutwert-Drehgebers mit dem Parameter Pn205 des Sigma-II-Servotreibers einstellen. Mit diesem Parameter wird die maximale Anzahl an Umdrehungen

eingestellt, die der Zähler zählt, bevor er einen Überlauf hat. Weitere Informationen zum Servotreiber-Parameter Pn205 finden Sie im Sigma-II-Servotreiber-Handbuch. Unter Berücksichtigung dieses Parameterwertes ist der maximale Positionswert, den der Drehgeber signalisieren kann, folgender:

$$max_drehgeber_impuls_wert = (Pn205 + 1) \cdot drehgeber_impulse - 1$$

Wodurch sich Pn205 vollständige Umdrehungen ergeben, plus die Position innerhalb einer Umdrehung (der Bruchteil zwischen 0 und 1 außer 1). Wenn die MECHATROLINK-Verbindung mit dem Antrieb hergestellt ist, wird die Position des Absolutwert-Drehgebers aus dem Antrieb gelesen und der Wert in **MPOS** geschrieben (nach der Umwandlung: **UNITS** × Pn202/Pn203). Wenn das mechanische System einen begrenzten Verfahrweg zurückzulegen hat, z. B. in einer Kugelumlaufspindel, sollte der Wert für den Parameter Pn205 groß genug eingestellt werden, dass ein Überlauf des Zählers aus der effektiven Position gegeben ist. Dies wird als begrenzte Achse oder finite Achse bezeichnet. Ein typisches Beispiel für eine begrenzte Achse ist eine Kugelumlaufspindel, wie in Abb.24 gezeigt. Wenn sich das mechanische System immer in die gleiche Richtung bewegt, wird der Überlauf des Multi-Umdrehungszählers erreicht. In diesem Fall muss der Wert von Pn205 gewährleisten, dass der Überlauf in Bezug auf die Maschine immer in derselben Position auftritt. Dies wird als unbegrenzte Achse bezeichnet; ein typisches Beispiel hierfür ist ein Drehtisch, wie in Abb.20 gezeigt. Dies kann mit der folgenden Gleichung erreicht werden: dem kleinsten Wert von *m*, so dass:

$$n \cdot Maschinen_zyklen = m \cdot Motor_umdrehung$$

Weil *n* und *m* Ganzzahlen sind: Pn205 = *m* – 1. Diese Einstellung wird im folgenden Beispiel erklärt.

Beispiel 2

Das mechanische System besteht aus einem einfachen Drehtisch, wie in der Abbildung gezeigt. Es wird ein Servomotor mit 16-Bit-Absolutwert-Drehgeber verwendet. Das Übersetzungsverhältnis des Getriebes ist 1:10. Die gewünschte Benutzereinheit ist Grad. Der Drehtisch ist in sechs Abschnitte von je 60 Grad unterteilt. Deshalb ist der Maschinen_zyklus 60 Grad.
Wenn wir die letzte Gleichung auf die obige Situation anwenden, ergibt sich:

$$10 \cdot \text{Motor_umdrehung} = 1 \cdot \text{Maschinen_umdrehung} = 6 \cdot \text{Maschinen_zyklus}$$

Diese Gleichung wird vereinfacht:

$$5 \cdot \text{Motor_umdrehung} = 3 \cdot \text{Maschinen_zyklus}$$

Das Ergebnis ist wie folgt:

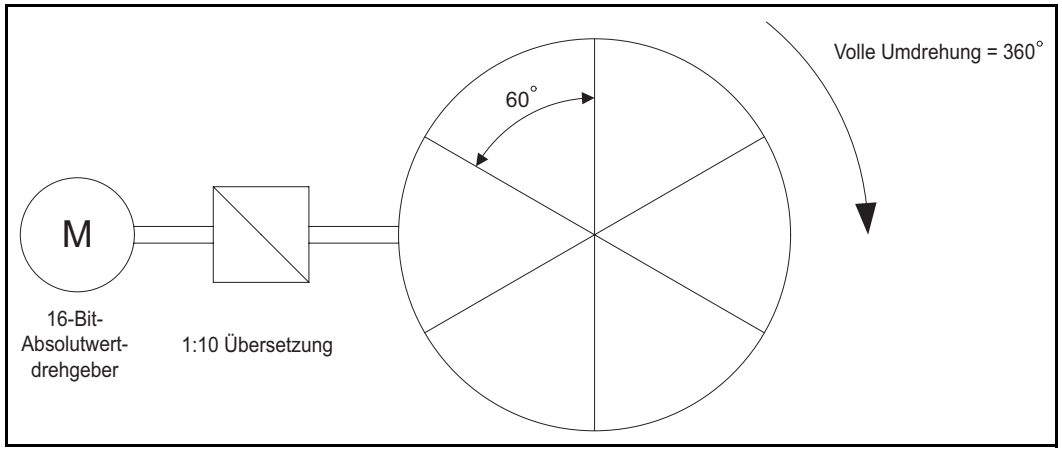
$$Pn205 = 5–1 = 4$$

Wir berechnen die Parameter wie in Beispiel 1. Dadurch ergibt sich:

$$UNITS = 2^{11} = 2048$$
$$Pn202 = 32$$
$$Pn203 = 36$$

Um sowohl in Trajexia als auch im Servotreiber den richtigen Überlauf zu garantieren, müssen wir zwei weitere Parameter einstellen: **REP_DIST** = 60 und **REP_OPTION** = 1. Mit diesen Einstellungen bewirkt der Befehl **MOVE(35)** eine Drehung des Tisches um 35 Grad in positive Richtung. Der Bereich möglicher Werte für **MPOS** und **DPOS** ist von 0 bis 60 Grad.

Abb.20



Sie müssen den Absolutwert-Drehgeber initialisieren, bevor Sie ihn zum ersten Mal einsetzen, wenn die Batterie bei der Stromabschaltung leer ist und wenn die Einstellung für den Multi-Umdrehungsgrenzwert im Parameter Pn205 geändert wird. Die Initialisierung kann über das Display des Servotreibers oder mit dem Software-Tool erfolgen. Weitere Informationen zum Initialisieren des Absolutwert-Drehgebers finden Sie im Sigma-II-Servotreiberhandbuch.

Es ist möglich, den Multi-Umdrehungszähler zurückzusetzen, aber es ist nicht möglich, die Position innerhalb einer Umdrehung (dem Bruchwert zwischen 0 und 1 außer 1) zurückzusetzen. Um einen Offset von 0 einzustellen, verwenden Sie den Parameter Pn808. Weitere Informationen finden Sie im Handbuch für die NS115 MECHATROLINK-II-Schnittstellenbaugruppe.

Beim Hochfahren wird die Position des Absolutwert-Drehgebers vom Motor gelesen und mit folgender Umwandlung zu **MPOS** geschrieben:

- Für **MPOS**:

$$\text{Absolute_MPOS} = \text{abs_position_drehgeber} \cdot \frac{1}{\text{UNITS}} \cdot \frac{\text{Pn203}}{\text{Pn202}}$$

- Dies ist korrekt, wenn:

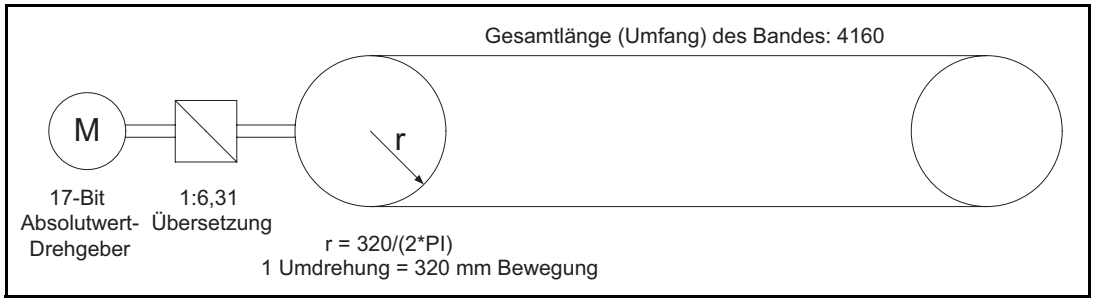
$$(\text{Pn205} + 1) \cdot \frac{\text{Pn203}}{\text{Pn202}} \cdot \text{drehgeber_impulse} < 5000000$$

- Wenn dieser Wert größer als 5.000.000 ist, kann **MPOS** falsche Werte beim Hochfahren haben. Um dieses Problem zu vermeiden, fügen Sie nach allen **UNITS**-Initialisierungen den Programmcode **DEFPOS = ENCODER/UNITS** hinzu.

Beispiel 3

Das mechanische System verwendet einen Servomotor mit 17-Bit-Absolutwert-Drehgeber. Das mechanische Übersetzungsverhältnis des Getriebes ist 1:6,31. Eine Drehung der Riemenscheibe bewegt das bewegliche Teil des Bandes um 320 mm. Die Gesamtlänge des Bandes und damit der Gesamt-Bewegungsbereich des beweglichen Teils beträgt 4160 mm. Die mechanische Maßeinheit muss in mm angegeben werden. Das heißt, dass alle Achsenparameter und Befehle, die an Trajexia übermittelt werden, in mm ausgedrückt werden. Bei Verwendung der gleichen Prozedur wie in Beispiel 1 sieht die Gleichung, die die Beziehung zwischen Benutzereinheiten und Drehgeberimpulsen ausdrückt, wie folgt aus:

Abb.21



$$\frac{Pn202}{Pn203}\,UNITS = \frac{2^{17}\cdot Drehgeber_impulse}{1\cdot Motor_umdrehung}\,\frac{6{,}31\cdot Motor_umdrehung}{1\cdot Riemenscheiben_umdrehung}\,\frac{1\cdot Riemenscheiben_umdrehung}{320\ mm} =$$

$$\frac{2^{17}\cdot 6{,}31}{320}\,\frac{Drehgeber_impulse}{mm}$$

Deshalb gilt:

$$\frac{Pn202}{Pn203}\,UNITS = \frac{2^{17}\cdot 6{,}31}{320} = \frac{2^{17}}{2^5}\frac{631}{1000} = 2^{12}\,\frac{631}{8{,}125} = 2^{12}\,\frac{631}{2^3\cdot 125} = 2^9\,\frac{631}{125}$$

Eine Lösung ist:

$UNITS = 2^9 = 512$
$Pn202 = 631$
$Pn203 = 125$

Bitte beachten Sie, dass wir den Radius der Riemenscheibe in der Berechnung nicht verwendet haben. Der Grund ist, die Verwendung von π zu vermeiden, weil diese Größe nicht als Bruchzahl ausgedrückt werden kann). Bei Zahnriemenscheiben wird häufig die Anzahl der Zähne und der mm pro Zahn verwendet.

Die Berechnung der Einstellung für
den Multi-Umdrehungsgrenzwert ist:

$$m \cdot \text{Motor_umdrehung} = n \ \text{Maschinen_zyklus}$$

$$m \cdot \text{Motor_umdrehung} = n \ \text{Maschinen_zyklus} \ \frac{4160 \cdot \text{Riemenscheiben_umdrehung}}{320 \cdot \text{Maschinen_zyklus}} = n \cdot 13 \ \text{Riemenscheiben_umdrehung}$$

$$= n \cdot 13 \ \frac{6 \cdot 31 \ \text{Motor_umdrehung}}{1 \ \text{Riemenscheiben_umdrehung}} = n \cdot 82{,}03 \ \text{Riemenscheiben_umdrehung}$$

$$m = n \cdot 82{,}03$$

Die kleinste Ganzzahl *m*, für die diese Gleichung gilt, ist 8203.
Daher ist Pn205 = 8202.
Um darüber hinaus den Bewegungseinheitenbereich
auf den sich bewegenden Bereich des beweglichen Teils
zu beschränken, müssen die folgenden Achsenparameter
gesetzt werden: **REP_DIST = 4260** und **REP_OPTION = 1**.
Mit diesen Einstellungen bewirkt die Ausführung von
MOVE(38) eine Bewegung des beweglichen Teils
von 38 mm in Vorwärtsrichtung. Der Bereich möglicher
Werte für **MPOS** und **DPOS** ist von 0 mm bis 4160 mm.

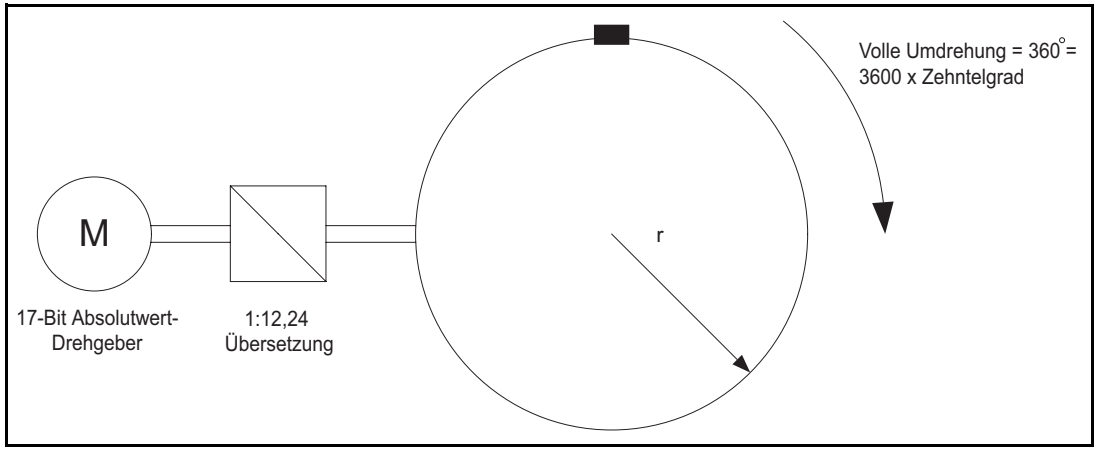
Beispiel 4

Das mechanische System verwendet einen Servomotor
mit 17-Bit-Absolutwert-Drehgeber. Das mechanische
Übersetzungsverhältnis des Getriebes ist 1:12,24.
Die mechanische Maßeinheit muss in Zehntelgrad angegeben
werden. Deshalb beträgt der Gesamt-Wiederholungsabstand
für die volle Umdrehung des beweglichen Teils 3600 Zehntelgrad.
Mit der gleichen Prozedur wie in Beispiel 1 ergibt sich:

$$\frac{\text{Pn202}}{\text{Pn203}} \ \text{UNITS} = \frac{2^{17} \cdot \text{Drehgeber_impulse}}{1 \cdot \text{Motor_umdrehung}} \ \frac{12{,}24 \cdot \text{Motor_umdrehung}}{1 \cdot \text{Maschinen_umdrehung}} \ \frac{1 \cdot \text{Riemenscheiben_umdrehung}}{3600 \ \text{Zehntelgrad}} =$$

$$= \frac{2^{17} \cdot 12{,}24}{3600} \ \frac{\text{Drehgeber_impulse}}{\text{Zehntelgrad}}$$

Abb.22



Deshalb gilt:

$$UNITS = \frac{Pn202}{Pn203} = 2^{17}\,\frac{1224}{360000}$$

Eine Lösung ist:

$$UNITS = 2^{17} = 131072$$
$$Pn202 = 1224$$
$$Pn203 = 360000$$

Weil der größte gemeinsame Teiler von Pn202 und Pn203 1 sein muss, erhalten wir: Pn202 = 17 und Pn203 = 500. Daher sind die Parameter:

UNITS = 131072
Pn202 = 17
Pn203 = 500
Pn205 = 16
REP_DIST = 3600
REP_OPTION = 1

Um die Einstellung für den Multi-Umdrehungsgrenzwert Pn205 zu berechnen, haben wir:

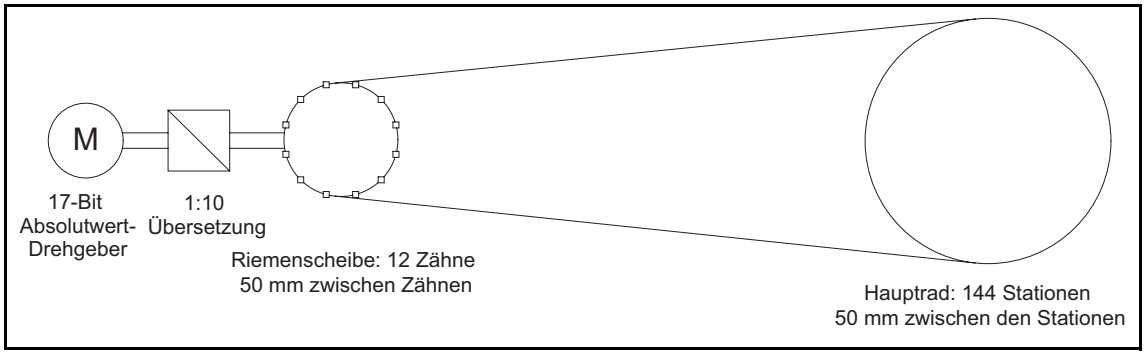
$$m \cdot Motor_umdrehung = n \cdot Maschinen_zyklus = n \cdot 12{,}24 \cdot Motor_umdrehung$$

Die Lösung ist: *n* = 100 und *m* = 1224. Wenn wir die Faktoren vereinfachen, ergibt sich: *n* = 25 und *m* = 306. Deshalb: Pn205 = m – 1 = 305. Mit diesen Einstellungen bewirkt die Ausführung von **MOVE(180)** eine Bewegung des beweglichen Teils von 180 Zehntelgrad oder 18 Grad in Vorwärtsrichtung.

Beispiel 5

Das mechanische System verwendet einen Servomotor mit 17-Bit-Absolutwert-Drehgeber. Das mechanische Übersetzungsverhältnis des Getriebes ist 1:10. Die Riemenscheibe hat 12 Zähne, die jeweils 50 mm auseinander liegen. Eine vollständige und Drehung der Riemenscheibe entspricht 144 Stationen am Hauptrad. Der Abstand zwischen zwei Stationen beträgt 50 mm. Die mechanische Maßeinheit muss mm sein.
Der Gesamt-Wiederholungsabstand muss der Abstand zwischen zwei Stationen sein, nämlich 50 mm.
Mit der gleichen Prozedur wie in Beispiel 1 ergibt sich:

Abb.23



$$\frac{Pn202}{Pn203}\ \text{UNITS} =$$

$$\frac{2^{17}\cdot \text{Drehgeber_impulse}}{1\cdot \text{Motor_umdrehung}} \cdot \frac{10\cdot \text{Motor_umdrehung}}{1\cdot \text{Riemenscheiben_umdrehung}} \cdot \frac{1\cdot \text{Riemenscheiben_umdrehung}}{12\cdot \text{Station}} \frac{1\cdot \text{Station}}{50\ \text{mm}} =$$

$$= \frac{2^{17}\cdot 10}{12\cdot 50} \frac{\text{Drehgeber_impulse}}{\text{mm}}$$

Wenn wir daher das mechanische System zur Einstellung des elektronischen Übersetzungsverhältnisses benutzen, haben wir:

$$\text{UNITS}\ \frac{Pn202}{Pn203} = \frac{2^{17}}{50}\frac{10}{12}$$

Eine mögliche Lösung ist:

$$\text{UNITS} = \frac{2^{17}}{50}$$

Pn202 = 5
Pn203 = 6
Pn205 = 4

Weil $2^{17}/50$ eine Zahl mit einer unendlichen Anzahl von Nachkommastellen ist, können wir Folgendes wählen:

$$\text{UNITS}\ \frac{Pn202}{Pn203} = 2^{17}\frac{10}{50\cdot 12} = 2^{17}\frac{10}{600} = 2^{17}\frac{1}{60} = 2^{17}\frac{1}{2^2\cdot 15} = 2^{15}\frac{1}{15}$$

Daher sind die Parameter wie folgt:

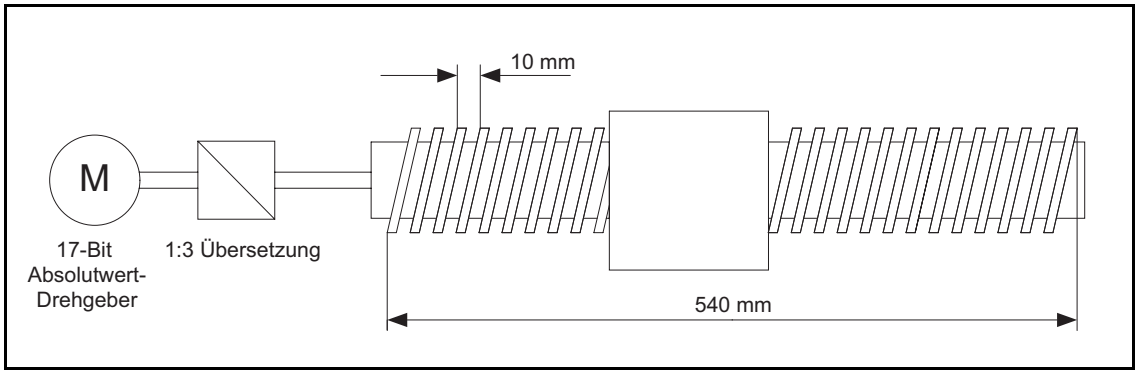
UNITS = 2^{15} = 32768
Pn202 = 1
Pn203 = 15
Pn205 = 4
REP_DIST = 50
REP_OPTION = 1

Mit diesen Einstellungen bewirkt die Ausführung von **MOVE(50)** eine Bewegung des beweglichen Teils von 50 mm oder eine Station in Vorwärtsrichtung.

Beispiel 6

Das mechanische System besteht aus einer Kugelspindel. Es wird ein Servomotor mit 17-Bit-Absolutwert-Drehgeber verwendet.
Das mechanische Übersetzungsverhältnis des Getriebes ist 1:3.
Die Steigung der Kugelspindel beträgt 10 mm pro Umdrehung.
Der Gesamt-Verfahrweg der Kugelspindel beträgt 540 mm.
Die mechanische Maßeinheit muss mm sein.
Mit der gleichen Prozedur wie in Beispiel 1 ergibt sich:

Abb.24



$$\frac{Pn202}{Pn203}\,UNITS = \frac{2^{17}\cdot Drehgeber_impulse}{1\cdot Motor_umdrehung}\,\frac{3\cdot Motor_umdrehung}{1\cdot Kugelspindel_umdrehung}\,\frac{1\cdot Kugelspindel_umdrehung}{10\ mm} =$$

$$= \frac{2^{17}\cdot 3}{10}\,\frac{Drehgeber_impulse}{mm}$$

Deshalb gilt:

$$\frac{Pn202}{Pn203}\,UNITS = 2^{17}\,\frac{3}{10} = 2^{17}\,\frac{3}{2\cdot 5} = 2^{16}\,\frac{3}{5}$$

Eine Lösung ist:

UNITS = 2^{16} = 65536
Pn202 = 3
Pn203 = 5

Die Berechnung der Einstellung für den Multi-Umdrehungsgrenzwert-Parameter Pn205 wird in diesem Fall nicht benötigt, weil die Kugelspindel ein System mit einer finiten (begrenzten) Achse ist. Es reicht aus, diesen Wert hoch genug zu setzen, so dass der Überlauf des Zählers außerhalb der effektiven Position stattfindet. Weil es sich um eine finite Achse handelt, ist es nicht wichtig, den Parameter **REP_OPTION** einzustellen, weil **REP_DIST** so groß eingestellt werden muss, dass er außerhalb der maximalen effektiven Position liegt (540 mm). Eine Lösung ist: **REP_DIST** = 1000 und **REP_OPTION** = 0.
Mit dieser Einstellungen bewirkt die Ausführung von **MOVE(17)** eine Bewegung der Kugelspindel von 17 mm in Vorwärtsrichtung.

6.1.4 Zuordnung der Servotreiber-Eingänge und -Ausgänge

Der Trajexia-Controller hat eine digitalen E/A-Einheit, die aus 256 Digitaleingängen und 256 Digitalausgängen besteht.
Der Digitalausgangsbereich hat vier Teile:

- Digitalausgänge 0–7.
  Diese Ausgänge existieren physikalisch nicht im TJ1-MC__.
  Wenn Sie zu diesen Ausgängen schreiben, hat dies keine Auswirkung.
  Wenn Sie diese Ausgänge auslesen, wird 0 zurückgegeben.
- Digitalausgänge 8–15.
  Diese Ausgänge existieren physikalisch im TJ1-MC__. Sie sind physikalisch über die schraublosen 28-Pin-Anschlussstecker an der Vorderseite des TJ1-MC__ zugänglich (weitere Informationen finden Sie im Hardware-Referenzhandbuch). Wenn Sie zu diesen Ausgängen schreiben, werden sie aktiv und geben ein 24 V DC-Signal aus. Wenn Sie diese Ausgänge auslesen, wird ihr aktueller Status zurückgegeben. Verwenden Sie den Befehl **OP**, um diese Ausgänge auszulesen oder zu ihnen zu schreiben.
- Digitalausgänge 16–31.
  Die Ausgänge sind nur Software-Ausgänge. Sie existieren physikalisch nicht im TJ1-MC__, aber Sie können zu ihnen schreiben und ihren richtigen Status auslesen. Diese Ausgänge werden meistens in BASIC-Programmen verwendet, um bestimmte Steuerungssequenzen durchzuführen, die Ausgänge erfordern, die nicht physikalisch zu existieren brauchen. Verwenden Sie den Befehl **OP**, um diese Ausgänge auszulesen oder zu ihnen zu schreiben.
- Digitalausgänge 32–255.
  Diese Ausgänge existieren physikalisch nur dann, wenn zusätzliche digitale E/A-Baugruppen über einen MECHATROLINK-II-Bus an den TJ1-MC__ angeschlossen werden. Werden sie ausgelesen oder wird zu ihnen geschrieben, wenn sie nicht physikalisch existieren (wenn die E/A-Baugruppen nicht angeschlossen sind), hat dies keine Auswirkung. Verwenden Sie den Befehl **OP**, um diese Ausgänge auszulesen oder zu ihnen zu schreiben.

Alle Ausgänge sind für den Controller eindeutig identifizierbar.
Auf sie wird nicht über die Achsen zugegriffen.

Der Digitaleingangsbereich hat drei Teile:

- Digitaleingänge 0–15.
  Diese Eingänge existieren physikalisch im TJ1-MC__. Sie sind physikalisch über die schraublosen 28-Pin-Anschlussstecker an der Vorderseite des TJ1-MC__ zugänglich (weitere Informationen finden Sie im Hardware-Referenzhandbuch). Diese Eingänge sind aktiv (ON), wenn ein 24 V DC-Signal auf sie angewendet wird. Wenn Sie sie auslesen, wird ihr aktueller Status zurückgegeben. Verwenden Sie den Befehl **IN**, um diese Eingänge auszulesen.
- Digitaleingänge 16–31.
  Diese Eingänge sind nur Software-Eingänge. Sie existieren physikalisch nicht im TJ1-MC__, aber Sie können Sie auslesen. Sie werden meistens in BASIC-Programmen verwendet, um bestimmte Steuerungssequenzen durchzuführen, die Eingänge erfordern, die nicht physikalisch zu existieren brauchen. Verwenden Sie den Befehl **IN**, um diese Eingänge auszulesen.
- Digitaleingänge 32–255.
  Diese Eingänge existieren physikalisch nur dann, wenn zusätzliche digitale E/A-Baugruppen über einen MECHATROLINK-II-Bus an den TJ1-MC__ angeschlossen werden. Werden sie ausgelesen oder wird zu ihnen geschrieben, wenn sie nicht physikalisch existieren (wenn die E/A-Baugruppen nicht angeschlossen sind), wird 0 zurückgegeben. Verwenden Sie zum Auslesen dieser Eingänge den Befehl **IN**.

Alle Eingänge sind für den Controller eindeutig identifizierbar.
Auf sie wird nicht über die Achsen zugegriffen.

MECHATROLINK-II-Servotreiber-Eingänge in der Trajexia-E/A-Einheit

Mit dem BASIC-Befehl **IN** können Sie auf die physikalisch existierenden Eingänge in einem BASIC-Programm zugreifen. Diese Eingänge können im Controller installiert oder über den MECHATROLINK-II-Bus angeschlossen werden.
Einige Eingänge des Servotreibers werden in der Trajexia-E/A-Einheit zugewiesen. Deshalb können Sie auf diese Eingänge über das BASIC-Programm zugreifen. Trajexia unterstützt dies nur für Servotreiber, die über den MECHATROLINK-II-Bus an das Trajexia-System angeschlossen sind.
Die Auswahl der Eingänge des Servotreibers, die in der Trajexia-E/A-Einheit zugewiesen sind, hängt von dem Wert des Parameters Pn81E ab.
Der empfohlene Wert für Pn81E ist 0x4321.
Mit diesem Standardwert ist die Zuordnung für den Sigma-II-Servotreiber wie folgt:

DRIVE_MONITOR-Bit	Anschluss-Signal	Trajexia-Eingang
6	CN1-44	-
7	CN1-45	-
8	CN1-46	-
12	CN1-40	28
13	CN1-41	29
14	CN1-42	30
15	CN1-43	31

Mit diesem Standardwert ist die Zuordnung für den Junma-Servotreiber wie folgt:

DRIVE_MONITOR-Bit	Anschluss-Signal	Trajexia-Eingang
2	CN1-2	26
6	CN1-1	27

Der Zugriff auf Servotreiber-Eingänge, die auf diese Weise in der Trajexia-E/A-Einheit zugewiesen werden, erfolgt innerhalb des Programms pro Achse und nicht auf die übliche Weise über den Befehl **IN**. Die einzige Methode, wie Sie diese Eingänge im Programm verwenden können, besteht darin, sie den Achsenparametern **DATUM_IN**, **FHOLD_IN**, **FWD_IN** und **REV_IN** zuzuordnen. Welche Eingänge des Servotreibers für die Achse verwendet werden, hängt von der Achse ab, für die die Parameter gesetzt sind.

Beispiel: Wir haben zwei Sigma-II-Treiber, die den Controller-Achsen 0 und 3 zugeordnet sind. Für beide Achsen wollen wir das Eingangssignal CN1-41 verwenden, das als Rückwärts-Begrenzungseingang dienen soll. Dies kann mit folgenden Befehlen erfolgen:

```
REV_IN AXIS(0) = 29
REV_IN AXIS(3) = 29
```

Bitte beachten Sie: Obwohl die Parameter **REV_IN** für beide Achsen den gleichen Wert 29 haben, sind die verwendeten realen Eingänge nicht die gleichen. Für Achse 0 wird Eingang CN1-41 des ersten Treibers (der dieser Achse zugeordnet ist) benutzt, aber für Achse 3 wird der gleiche Eingang CN1-41 des anderen Treibers (der Achse 3 zugeordnet ist) benutzt. Deshalb sagen wir, dass der Zugriff auf diese Eingänge pro Achse erfolgt; sie sind nicht für den gesamten Controller eindeutig. Im Allgemeinen haben diese beiden Eingänge gleichzeitig einen unterschiedlichen Status. Bitte beachten Sie auch, dass auf keinen der beiden Eingänge über den Befehl **IN** zugegriffen werden kann. Beispielsweise gibt der Befehl **IN(29)** den Status des Controller-Softwareeingangs 29 (eindeutig für alle Achsen) zurück, der einen anderen Status hat als Servotreiber-Eingänge, die derselben Nummer zugeordnet sind.
Weitere Informationen über die Zuordnung von MECHATROLINK-II-Servotreiber-Eingängen in der Trajexia-E/A-Einheit finden Sie im Trajexia-Programmierhandbuch unter den BASIC-Befehlen **DATUM_IN**, **FHOLD_IN**, **FWD_IN** und **REV_IN**.

6.1.5 Nullpunktsuche

Die Nullpunktsuche- oder Nullpunktsuch-Funktionalität wird oft als besondere Folge von Bewegungen einer Achse beim Hochfahren der Maschine gesehen. Diese Sequenz wird in den meisten Fällen automatisch ausgeführt, ohne Eingabe des Maschinenbedieners. Grundsätzlich wird bei einem Nullpunkt-Suchvorgang eine Position an eine spezifische Achse gekoppelt. Der Vorgang ist abhängig von den verwendeten Drehgebern (absolut oder relativ), vom verwendeten System (linear oder kreisförmig) und von der mechanischen Konstruktion der Maschine. Absolutwert-Drehgeber brauchen keine Bewegung während des Nullpunkt-Suchvorgangs, weil die genauen Positionen direkt zum System übertragen werden. Bei anderen Drehgebertypen ist eine Bewegung notwendig, da die exakte Position innerhalb des Systems unbekannt ist. Grundsätzlich erfolgt diese Bewegung bei niedriger Drehzahl in eine Richtung, bis ein bestimmter Messpunkt erreicht ist. Ein solcher Messpunkt kann aus beiden Richtungen erfasst werden, um die Präzision zu erhöhen. Beim Hochfahren sind die aktuellen Positionen der Achsen bei der Verwendung von Inkremental-Drehgebern 0. Weil diese Positionen nicht mit dem mechanischen Nullpunkt der Maschine übereinstimmen, ist es notwendig, die Nullpunktsuch-Sequenz auszuführen. Wird ein Absolutwert-Drehgeber verwendet, wird die absolute Position beim Hochfahren vom Drehgeber ausgelesen, und eine Nullpunktsuche ist nicht notwendig. In diesem Fall muss eine Startsequenz einmal während der Inbetriebnahme der Maschine ausgeführt werden.
In der Praxis gibt es mehrere verschiedene Nullpunkt-Suchsequenzen. Sie unterscheiden sich in folgenden Bereichen:

- Durch die Mittel, die zur Erfassung von der Begrenzungspositionen des beweglichen Teils verwendet werden (Sensoren, Schalter usw.)
- Durch Nullpunkt- (Ausgangs-) Position oder Sollwert.
- Durch mögliche Positionen des beweglichen Teils in Bezug auf Begrenzungspositionen und Nullpunktposition.

Trajexia enthält einige vordefinierte einfache Nullpunktsuch -Sequenzen:

- **DATUM(0)**
  Dies ist eigentlich keine Nullpunktsuche. Dieser Befehl stellt den Parameter **DPOS=MPOS** und löscht die Achsenfehler.
- **DATUM(1)**
  Dies ist eine Nullpunktsuche in Vorwärtsrichtung unter Verwendung der Z-Marke eines Drehgebers als Nullpunkt-Schalter.
- **DATUM(2)**
  Dies ist eine Nullpunktsuche in Rückwärtsrichtung unter Verwendung der Z-Marke eines Drehgebers als Nullpunkt-Schalter.
- **DATUM(3)**
  Dies ist eine Nullpunktsuche in Vorwärtsrichtung unter Verwendung des Eingangs, der im Parameter **DATUM_IN** als Nullpunkt-Schalter ausgewählt wurde.
- **DATUM(4)**
  Dies ist eine Nullpunktsuche in Rückwärtsrichtung unter Verwendung des Eingangs, der im Parameter **DATUM_IN** als Nullpunkt-Schalter ausgewählt wurde.
- **DATUM(5)**
  Dies ist eine Nullpunktsuche in Vorwärtsrichtung unter Verwendung des Eingangs, der im Parameter **DATUM_IN** als Nullpunkt-Schalter ausgewählt wurde; es wird die nächste Z-Marke eines Drehgeber gesucht.
- **DATUM(6)**
  Dies ist eine Nullpunktsuche in Rückwärtsrichtung unter Verwendung des Eingangs, der im Parameter **DATUM_IN** als Nullpunkt-Schalter ausgewählt wurde; es wird die nächste Z-Marke eines Drehgebers gesucht.

Weitere Informationen über diese vordefinierten Nullpunktsuch-Sequenzen finden Sie in Abschnitt 3.2.68.
In einigen Situationen sind komplexere Nullpunktsuch-Sequenzen erforderlich:

- Absolutwertschalter-Nullpunktsuche plus Endschalter.
- Nullpunktsuche mit Hilfe eines Endschalters.
- Nullpunktsuche mit Hilfe einer mechanischen Begrenzung, die die Bewegung blockiert.
- Nullpunktsuche mit Drehgeber-Sollwertimpuls „Z-Marke“.
- Statische Nullpunktsuche, bei der eine Position über einen Benutzer-Sollwert erzwungen wird.
- Statische Nullpunktsuche, bei der eine Position über einen Absolutwert-Drehgeber erzwungen wird.

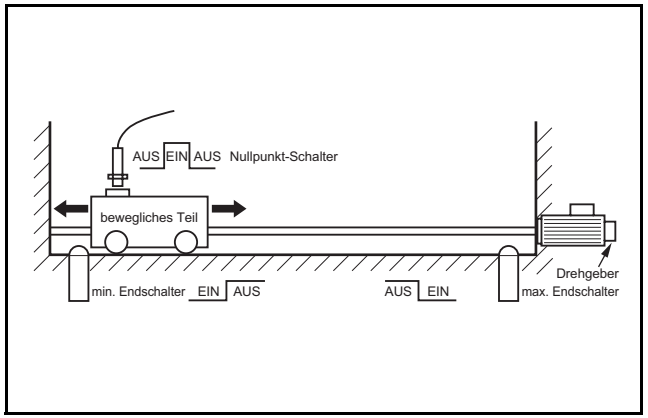
Die Abbildung zeigt ein allgemeines Nullpunktsuch-Szenario.
Diese einfache Nullpunktsuch-Sequenz besteht aus drei Schritten:

1. Suche nach einem Signal.
2. Suche nach einem weiteren Signal.
3. Bewegung der Achse zu einer vordefinierten Position.

Aus Sicherheitsgründen sind die Endschalter im Gutzustand geschlossen. Aus diesem Grund wird in dieser Abbildung und in den folgenden Abbildungen in diesem Abschnitt der niedrige Signalpegel als EIN und der hohe Signalpegel als AUS angezeigt.

Es ist wichtig, Folgendes zu beachten: Bevor eine Nullpunktsuch-Prozedur ausgeführt wird, ist es notwendig, die Achsenparameter **UNITS**, **REP_DIST** und **REP_OPTION** sowie die Servotreiber-Parameter Pn202, Pn203 und Pn205 ordnungsgemäß und in Übereinstimmung mit dem mechanischen System und den gewünschten Maßeinheiten einzustellen, die bei der Programmierung verwendet wurden. Diese Parameter haben einen Einfluss auf die Nullpunktsuche, besonders dann, wenn ein Absolutwert-Drehgeber verwendet wird. Weitere Informationen zur Einstellung dieser Parameter finden Sie in Abschnitt 6.1.2.

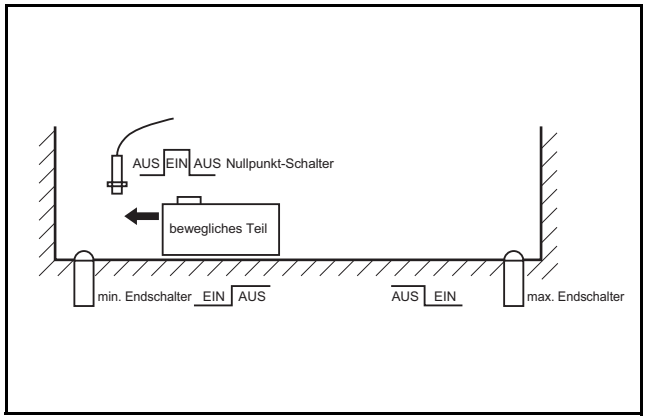
Abb.25



Absolutwertschalter-Nullpunktsuche plus Endschalter

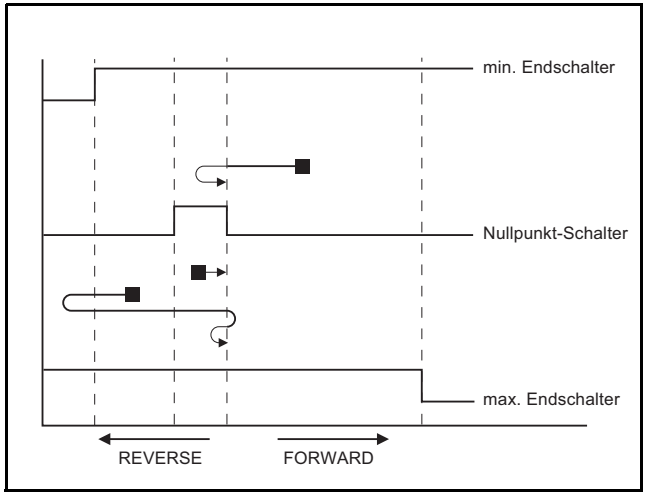
Die Nullpunkt-Suchfunktion wird ausgeführt, indem nach einem externen Endschalter gesucht wird, dessen Position den Nullpunkt definiert. Das Beispiel für diese Nullpunktsuch-Prozedur ist in der Abbildung zeigt.

Abb.26



Die Abbildung zeigt die möglichen Szenarios für die absolute Nullpunktsuche plus Endschalter. Diese Szenarios sind abhängig von der Position des beweglichen Teils, wenn die Versorgungsspannung eingeschaltet wird. Das Programmbeispiel, das diese Nullpunktsuchsequenz ausführt, ist unten angegeben.

Abb.27



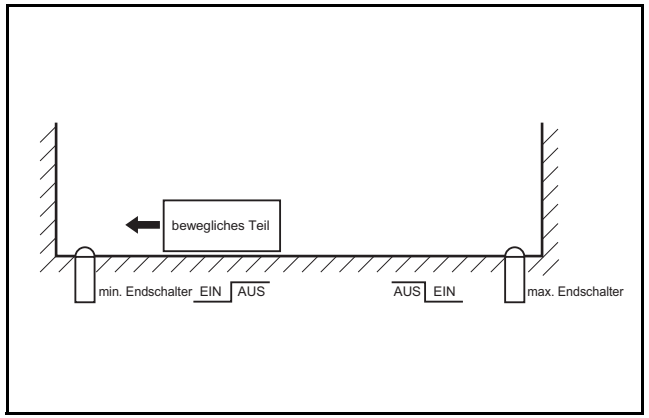

```
'Absoluter Nullpunktschalter: IN0
'Linker Endschalter: IN1
'Rechter Endschalter: IN2
BASE(0)
DATUM_IN=0
FW_IN=2
RV_IN=1
SERVO=ON
WDOG=ON
DATUM(4)
WA(1)
WAIT UNTIL MTYPE=0 OR IN(1)=OFF
IF IN(1)=ON
    FORWARD
    WAIT UNTIL IN(0)=ON
    WAIT UNTIL IN(0)=OFF
    CANCEL
```

```
    DATUM(4)
    WA(1)
    WAIT IDLE
ENDIF
```

Nullpunktsuche mit Hilfe von Endschaltern

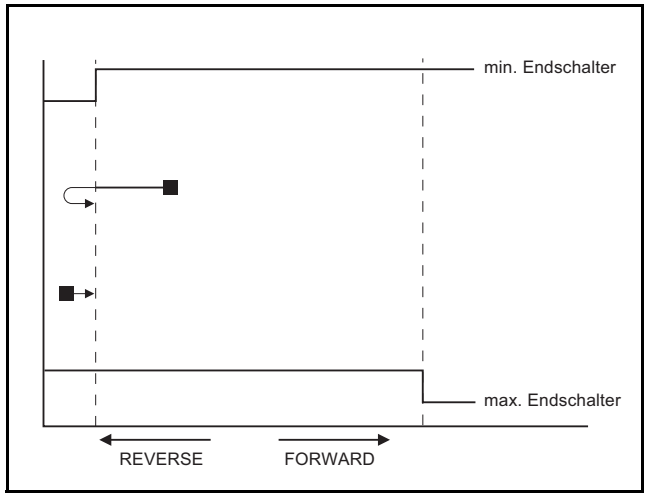
Diese Nullpunkt-Suchfunktion wird ausgeführt, indem nur nach externen Sensoren in Form von Endschaltern gesucht wird. Das Beispiel für diese Nullpunktsuch-Prozedur ist in der Abbildung zeigt.

Abb.28



Die Abbildung zeigt die möglichen Szenarios für eine Nullpunktsuche mit Hilfe von Endschaltern, abhängig von der Position des beweglichen Teils beim Einschalten der Versorgungsspannung
Das Programmbeispiel, das diese Nullpunktsuchsequenz ausführt, ist unten angegeben.

Abb.29

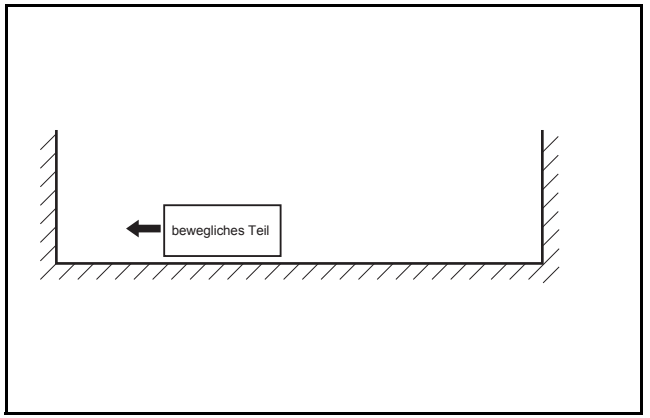



```
'Nullpunkt und linker Endschalter: IN0
'Rechter Endschalter: IN1
BASE(0)
DATUM_IN=0
SERVO=ON
WDOG=ON
DATUM(4)
WA(1)
WAIT IDLE
```

Nullpunktsuche gegen Hardwareteile, die die Bewegung blockieren

Diese Nullpunktsuch-Prozedur führt die Nullpunktsuche mit Hilfe einer mechanischen Begrenzung, die die Bewegung blockiert, durch. Es gibt keine Endschalter, keinen Absolut-Positionsschalter und keine Sollwertimpulse. Die Nullpunktposition wird erkannt, indem ein bestimmtes Drehmoment erkannt wird, das gegen die blockierenden Objekte ausgeübt wird. Es ist ein angemessener Drehmomentgrenzwert erforderlich, um die Mechanik während des Nullpunktsuchprozesses nicht zu beschädigen. Das Beispiel für diese Nullpunktsuch-Prozedur ist in der Abbildung zeigt.
Das Programmbeispiel, das diese Nullpunktsuchsequenz ausführt, ist unten angegeben.

Abb.30

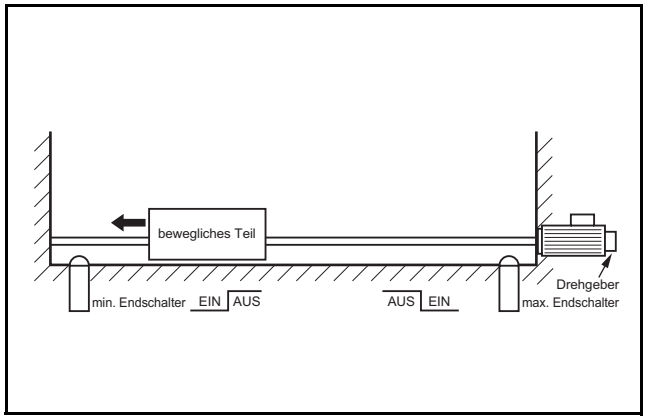



```
BASE(0)
DRIVE_CONTROL=11 'Drehmoment ueberwachen mit DRIVE_MONITOR
SERVO=ON
WDOG=ON
SPEED=CREEP
REVERSE
WA(1)
WAIT UNTIL DRIVE_MONITOR < -100
    'Auf bestimmten Betrag von ausgeuebtem Drehmoment warten
CANCEL
DEFPOS(0)
MOVEABS(10) 'Dies ist notwendig, da sonst die Position
            'bis zur Hardwaregrenze der Maschine
            'gehalten wird und der Motor bei Ueberlast
            abschaltet
```

Nullpunktsuche mit Drehgeber-Sollwertimpuls „Z-Marke“

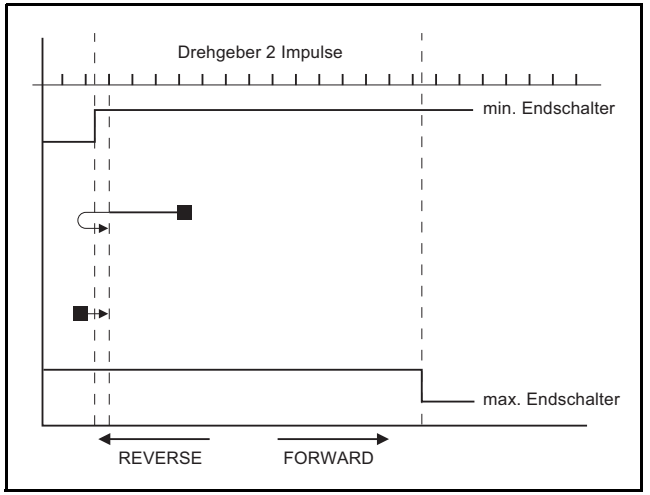
Bei dieser Nullpunktsuch-Prozedur wird nach dem „Z-Marken“-Signal des Drehgebers gesucht. Dieses Signal wird auch als „Z-Marke“ oder „Referenzimpuls“ bezeichnet. Es erscheint einmal bei jeder vollen Drehgeberumdrehung. Das Beispiel für diese Nullpunktsuch-Prozedur ist in der Abbildung zeigt.

Abb.31



Die Abbildung zeigt die möglichen Szenarios für eine Nullpunktsuche mit dem Drehgeber-Sollwertimpuls „Z-Marke“, abhängig von der Position des beweglichen Teils beim Einschalten der Versorgungsspannung.
Das Programmbeispiel, das diese Nullpunktsuchsequenz ausführt, ist unten angegeben.

Abb.32




```
'Nullpunkt und linker Endschalter: IN0
'Rechter Endschalter: IN1
REV_IN=-1
BASE(0)
DATUM_IN=0
SERVO=ON
WDOG=ON
DATUM(6)
WA(1)
WAIT IDLE
```

Statische Nullpunktsuche, bei der eine Position über einen Benutzer-Sollwert erzwungen wird

Diese Nullpunktsuch-Prozedur führt eine statische Nullpunktsuche durch, indem eine tatsächliche Position direkt erzwungen wird. Sie führt keine physikalische Bewegung durch.

```
DATUM(0)
```

Statische Nullpunktsuche, bei der eine Position über einen Absolutwert-Drehgeber erzwungen wird

Diese Nullpunktsuch-Prozedur setzt eine tatsächliche Position auf die Position eines Absolutwert-Drehgebers. Sie führt keine physikalische Bewegung durch. Sie ist nur bei einer Achse mit einem Absolutwert-Drehgeber in einem Regelkreis möglich.

6.1.6 Registrierung

Bei der Registrierung, auch als „Positionsspeichersignal“ oder „Druckmarkenerkennung“ bezeichnet, wird die Position einer Achse in Echtzeit gespeichert, wenn ein externer Eingang aktiviert wird. Die registrierten, d. h. gespeicherten Daten werden später (nicht in Echtzeit) durch das Anwendungsprogramm verarbeitet. Die Registrierung unterscheidet sich von der Verarbeitung eines Interrupt-Eingangs oder -Signals. Bei der Registrierung wird kein Ereignis erzeugt, wenn der Registrierungseingang aktiviert ist. Außerdem wird die normale Ausführung des Anwendungsprogramms nicht gestört oder unterbrochen. Nur die Position einer Achse wird gespeichert. Diese Daten können ebenso wie andere Parameter oder Werte in einem Programm benutzt werden. Die Registrierungsdaten sind unmittelbar nach der Registrierung für ein Programm verfügbar.

Der Vorteil der Registrierung besteht darin, dass sie sehr schnell erfolgt. Deshalb ist die Achsenposition, die gespeichert wird, sehr genau. Um diese Geschwindigkeit und Genauigkeit zu erreichen, ist die Registrierung in die Hardware implementiert; der Registrierungseingang muss sich auf derselben Karte wie der Drehgeber-Eingang befinden, der die Informationen über die Achsenposition bereitstellt.
Das Erfassen und Speichern der Achsenposition erfolgt in Echtzeit durch die Hardware. Die Verarbeitung dieser Daten erfolgt später (nicht in Echtzeit) durch das Anwendungsprogramm.

Der Achsenbefehl REGIST

In Trajexia wird eine Registrierung mit dem Achsenbefehl **REGIST** durchgeführt. Der Befehl nimmt ein Argument an. Dieses Argument bestimmt, welcher externe Eingang registriert wird, ob die Registrierung auf der steigenden oder fallenden Flanke des Eingangssignals ausgeführt wird, ob ein Erfassungsfenster verwendet wird sowie andere Optionen. Weitere Informationen zum Achsenparameter **REGIST** finden Sie in Abschnitt 3.2.219.
Die Registrierung ist bei verschiedenen Achsen unterschiedlich, je nachdem, wie sie an das System angeschlossen sind. Wenn eine Achse über den MECHATROLINK-II-Bus angeschlossen ist, erfolgt die Registrierung über die Servotreiber-Hardware. Wenn eine Achse über die Servotreiber-Analogschnittstelle und den TJ1-FL02 angeschlossen ist, erfolgt die Registrierung über die Hardware im TJ1-FL02.
Die unterschiedlichen Registrierungen sind unten beschrieben.

Registrierung im Sigma-II-Servotreiber

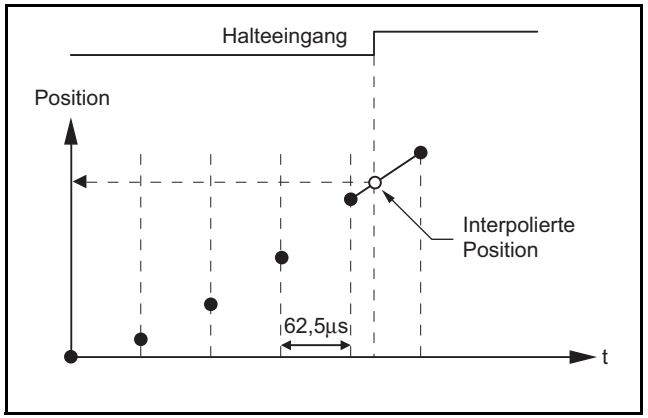
Die Registrierung im Sigma-II-Servotreiber erfolgt, wenn eine Achse, die dem Sigma-II-Servotreiber zugeordnet ist, über den MECHATROLINK-II-Bus an das Trajexia-System angeschlossen ist. Es gibt drei Registrierungseingänge am Sigma-II-Servotreiber Driver, aber nur ein Hardware-Positionsspeichersignal, deshalb kann nur ein Eingang gleichzeitig benutzt werden. Die physikalischen Eingänge befinden sich in den Pins CN1-44, CN1-45 und CN1-46 der 50-poligen-Steckverbindung CN1, aber Trajexia verwendet die Logikeingänge EXT1, EXT2 und EXT3, um die physikalischen Eingänge Logikeingängen zuzuordnen. Diese Zuordnung erfolgt über die Einstellung des Parameters Pn511 des Servotreibers. Weitere Informationen zur Einstellung dieser Zuordnung und den Parameter Pn511 finden Sie in Abschnitt 3.2.219, Tabelle 1.
Der für die Registrierung verwendete Eingang wird durch das Argument des Befehls **REGIST** bestimmt.

Die Verzögerung bei der Erfassung im Sigma-II-Servotreiber beträgt ungefähr 3 μs. Da die Drehgeberdaten alle 62,5 μs, aktualisiert werden, ist es notwendig, eine Interpolation vorzunehmen, um den richtigen Wert der erfassten Position zu erhalten (siehe Abbildung). Da sich die Motordrehzahl innerhalb von 62,5 μs kaum ändern kann, ist die Genauigkeit sehr hoch.
Die Verzögerungen bei der Übertragung der Daten sind:

- Verzögerung bei der Auslösung der Registrierung: 0,625 ms bis 4 ms.
- Verzögerung beim Empfang der Registrierung: 3,5 ms.
- Verzögerung bei der Erfassung der Registrierung: 3 μs.

Es ist auch möglich, die Z-Marke des Drehgebers zur Registrierung einer Achsenposition zu verwenden. Dies erfolgt ebenfalls mit dem Argument des Befehls **REGIST**.

Abb.33



Registrierung im Junma-Servotreiber

Die Registrierung im Junma-Servotreiber ist die gleiche wie die Registrierung im Sigma-II-Servotreiber, bis auf einen Unterschied: Es gibt nur einen physikalischen Eingang und ein logisches Positionsspeichersignal, deshalb sind keine Einstellungen von Servotreiber-Parametern notwendig. Der physikalische Eingang ist dem logischen Positionsspeichersignal EXT1 zugeordnet, und nur die steigende Flanke kann zur Registrierung verwendet werden.

Registrierung im TJ1-FL02

Der TJ1-FL02 hat zwei physikalische Registrierungseingänge und zwei Positionsspeichersignal-Schaltkreise pro Drehgebereingang, die unabhängig voneinander verwendet werden können. Deshalb können zwei unabhängige Registrierungseingänge gleichzeitig benutzt werden. Weitere Informationen über die Nutzung von Registrierungseingängen finden Sie in den Abschnitten 3.2.170, 3.2.171, 3.2.217, 3.2.218 und 3.2.219.
Die Verzögerung bei der Erfassung beträgt 0,5 μs. Da die Drehgeberposition kontinuierlich aus dem Leitungstreiber-Drehgebereingang gelesen wird, ist eine Interpolation nicht notwendig. Die Verzögerung bei der Übertragung der erfassten Daten beträgt nur einen **SERVO_PERIOD**-Zyklus.

Verwendung der Registrierung in Anwendungsprogrammen

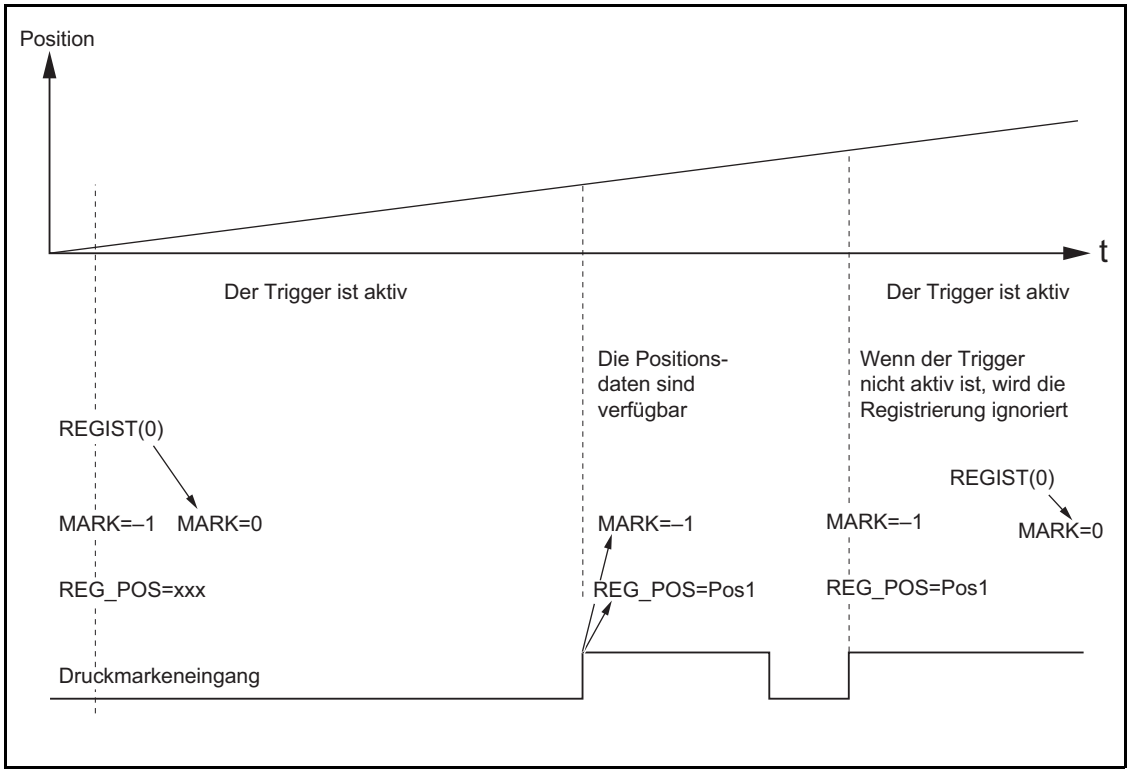
Es gibt einen Achsenbefehl (**REGIST**) und zwei Achsenparameter (**MARK** und **REG_POS**). Mit diesen Befehlen und Parametern können Sie die Registrierungsfunktionalität in BASIC-Programmen steuern und verwenden.

- Der Befehl **REGIST** erfasst die Achsenposition, wenn ein Druckmarkensignal erkannt wird. Die verfügbaren Einstellungen sind abhängig vom Achsentyp. Siehe Abschnitt 3.2.219.
- **MARK** ist ein Merker, der signalisiert, ob die Position erfasst wurde oder nicht. Für den zweiten Druckmarkeneingang des TJ1-FL02 ist außerdem der Parameter **MARKB** verfügbar. Weitere Informationen in den Sie in den Abschnitten 3.2.170 und 3.2.171.
- **REG_POS** hält die erfasste Achsenposition. Nur wenn der Merker **MARK** signalisiert, dass die Position erfolgreich erfasst wurde, kann davon ausgegangen werden, dass der Wert von **REG_POS** gültig ist. Für den zweiten Druckmarkeneingang des TJ1-FL02 ist außerdem der Parameter **REG_POSB** verfügbar. Weitere Informationen fin den Sie in den Abschnitten 3.2.217 und 3.2.218.

Die Abbildung zeigt die Abfolge der Befehlsausführungen und der Registrierungen des unten angegebenen Beispielprogramms.

```
BASE(N)
REGIST(0)
WAIT UNTIL MARK=0
schleife:
    WAIT UNTIL MARK=-1
    PRINT "Position erfasst in: "; REG_POS
    REGIST(0)
    WAIT UNTIL MARK=0
GOTO schleife
```

Abb.34

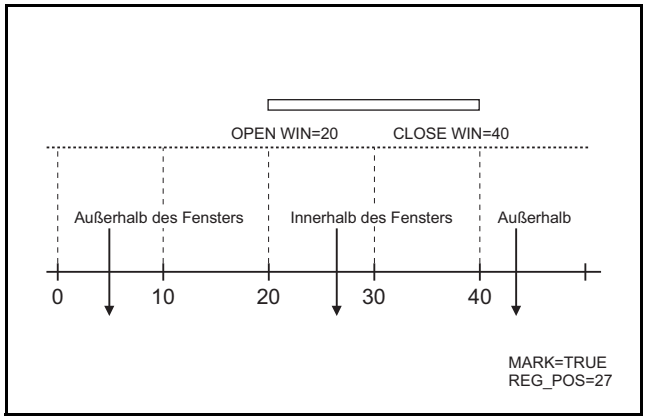


Druckmarken- und Erfassungsfenster

Das Erfassungsfenster ermöglicht es, dass Druckmarken nur innerhalb eines spezifizierten Bereichs von Achsenpositionen auftreten. Diese Funktion wird ausgewählt, indem der richtige Wert als Argument für den Befehl **REGIST** ausgewählt wird. Das Erfassungsfenster wird durch zwei Achsenparameter gesteuert, **OPEN_WIN** und **CLOSE_WIN**. Weitere Informationen zu den Parametern **REGIST**, **OPEN_WIN** und **CLOSE_WIN** finden Sie in den Abschnitten 3.2.49, 3.2.198 und 3.2.219.
Zwei Arten von Erfassungsfenstern stehen zur Verfügung:

- Das einschließende Erfassungsfenster akzeptiert Druckmarken nur innerhalb eines spezifizierten Fensters von Achsenpositionen. Bei dieser Variante werden Druckmarkenereignisse ignoriert, wenn die gemessene Position der Achse kleiner als der Achsenparameter **OPEN_WIN** oder größer als der Parameter **CLOSE_WIN** ist.
- Das ausschließende Erfassungsfenster akzeptiert Druckmarken nur außerhalb des spezifizierten Fensters von Achsenpositionen. Bei dieser Variante werden Druckmarkenereignisse ignoriert, wenn die gemessene Position der Achse größer als der Achsenparameter **OPEN_WIN** oder kleiner als der Parameter **CLOSE_WIN** ist.

Abb.35



Bei Verwendung des Erfassungsfensters ist der interne Prozess wie folgt:

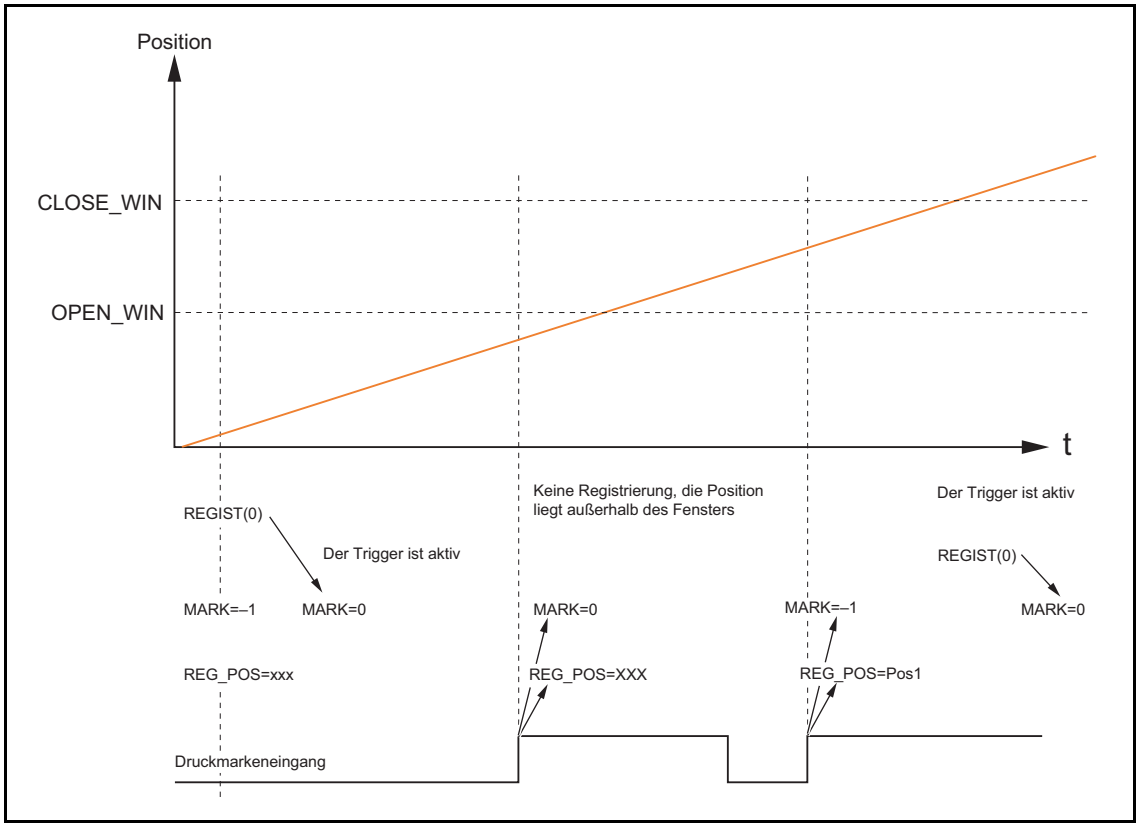
1. **REGIST** + Fenster wird im Programm ausgeführt.
2. **MARK** = 0, und das Positionsspeichersignal wird ausgelöst.
3. Die Position wird erfasst und an den Trajexia-Prozessor übertragen.
4. Ist die erfasste Position innerhalb des einschließenden Fensters oder außerhalb des ausschließenden Fensters?
   - Wenn ja, werden **MARK** = -1 und **REG_POS** aktualisiert.
   - Wenn nicht, wird zu Punkt 2 zurückgekehrt (das Positionsspeichersignal wird nochmals für den Benutzer ersichtlich ausgelöst).

Die Abbildung zeigt die Ausführungsabfolge der obigen Befehle und das Auftreten von Druckmarkenereignissen, wenn das einschließende Erfassungsfenster benutzt wird. Es gibt Verzögerungen zwischen diesen Ereignissen:

- Trajexia empfängt das Positionsspeichersignal.
- Trajexia beschließt, das Positionsspeichersignal nochmals auszulösen.
- Das Positionsspeichersignal wird ausgelöst.

Wegen dieser Verzögerungen gibt es eine gewisse Ungenauigkeit an den Flanken des Fensters, wenn Markierungen in der Nähe der Flanken erkannt werden. Auf Grund von Bus-Verzögerungen macht sich dies eher bemerkbar bei Achsen, die über den MECHATROLINK-II-Bus an das System angeschlossen sind. Um diese Verzögerungen auszugleichen, muss der Benutzer die Fensterränder groß genug wählen.

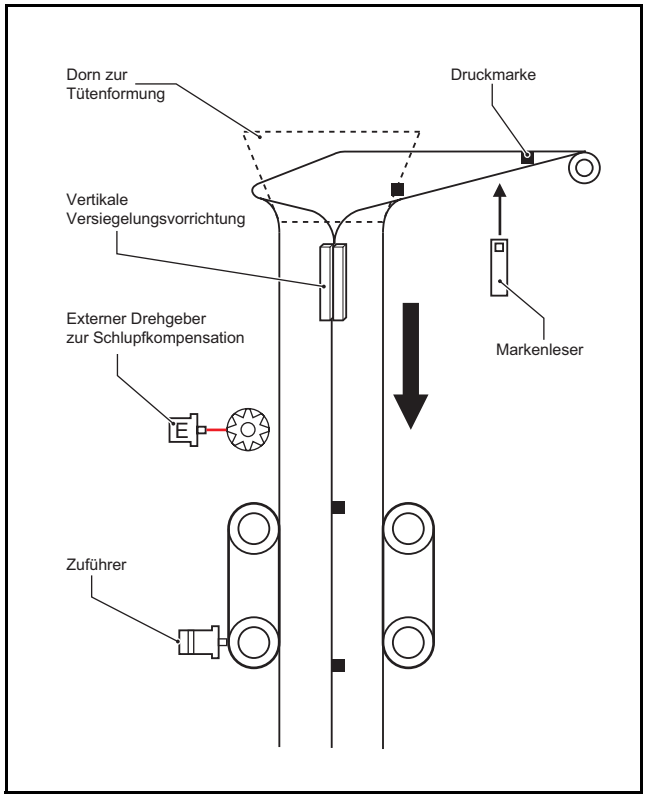
Abb.36



Beispiel: Korrektur einer Achsposition

Die Abbildung zeigt eine vertikale Befüllungs- und Versiegelungsmaschine zur Verpackung von Produkten in Tüten. Das Tütenmaterial wird von einer Kunststofffilmrolle zugeführt, die abgewickelt wird; dann wird sie über einen mechanischen Dorn zur Röhre eingeformt und die Röhre gleichzeitig vertikal versiegelt. Die Zuführbewegung erfolgt in Intervallen, und die Zufuhrlänge stimmt mit der Tütenlänge überein. Wenn die Tüte zugeführt ist, wird sie von der horizontalen Versiegelungseinrichtung verschlossen, sodass sie mit dem Produkt befüllt werden kann. Anschließend beginnt der Prozess von vorne, und es wird eine neue Tüte zugeführt.

Abb.37



Für den Zuführer stehen zwei Betriebsarten zur Verfügung: ohne Druckmarke und mit Druckmarke. Die Funktionsweise ohne Druckmarke ist eine einfache Punkt-zu-Punkt-Inkrementalbewegung. In diesem Fall gibt es keine Garantie dafür, dass der Zuführer genau die gleiche Strecke zurücklegt wie das Design-Muster. Nehmen wir zum Beispiel an, die Tütenlänge, die zugeführt werden muss, beträgt 200 mm, aber das tatsächliche Muster beträgt 200,1 mm. Bei einer einfachen Punkt-zu-Punkt-Inkrementalbewegung ohne Korrektur akkumuliert sich ein Fehler von 0,1 mm pro Tüte. Bei einer kleinen Anzahl Tüten ist der Unterschied nicht sichtbar, aber nach 500 Tüten beträgt der Fehler 50 mm, was 25 % der Tütenlänge entspricht.

Beim Einsatz von Druckmarken führt der Motion Controller eine Inkrementalbewegung bis zu einer bestimmten Position aus. Wird während der Positionierung die Druckmarke erkannt, wird die Zielposition „on-the-fly“ geändert, um die Bewegung an einer definierten Position nach der Druckmarke zu beenden. Deshalb ist in Bezug auf die Druckmarke immer die gleiche Strecke gewährleistet.

Abb.38

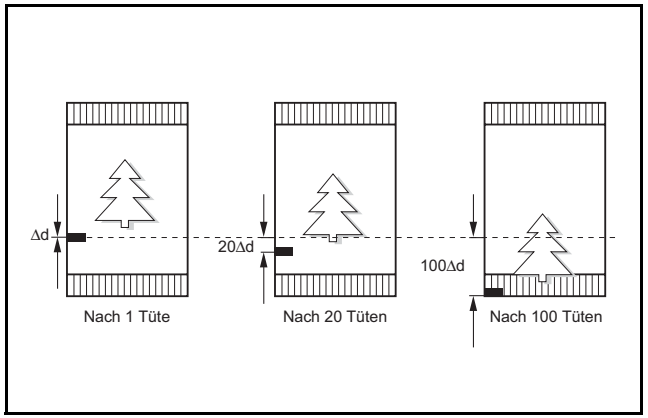
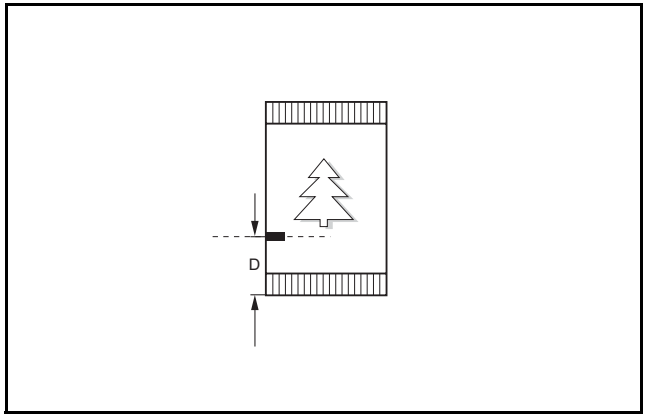


Abb.39



Das Bewegungsprofil und dessen auf Grund der Druckmarke vorgenommene Korrektur sind in Abb.39 gezeigt.
Das BASIC-Programm für dieses Beispiel ist:

```
DEFPOS(0)
REGIST(3)           'Druckmarke ausloesen
MOVE(tueten_laenge) 'Bewegung ueber theoretische Strecke
WA(1)
WAIT UNTIL MARK OR MTYPE=0
IF MARK THEN
    end_position=REG_POS+strecke_nach_marke
    MOVEMODIFY(end_position)
    'Korrektur der Strecke gemaess der Marke
ENDIF
```

Beispiel: Starten einer Slave-Achse an der genaue Position einer Master-Achse

Das Bild zeigt eine Schneidvorrichtung, die die „Kopfseite“ von Holztischen schneidet. Wenn das Holz zugeführt wird, wird die Holzkante von einer Fotozelle erkannt. Dann beginnt die Synchronisierung der Bewegung der Schneidvorrichtung mit der Position auf dem Holz.
Falls nach der Erkennung eines Signals von der Fotozelle die Bewegung vom Programm gestartet wird, gibt es immer mindestens eine **SERVO_PERIOD** (Servo Zykluszeit) lang Ungewissheit. Stattdessen wird die Bewegung über den Befehl **MOVELINK** mit **Synchronisierungs_option**=1 gestartet, was bewirkt, dass die Synchronisierung mit der Master-Achse beginnt, wenn das Registrierungsereignis bei der synchronisierten (Master-) Achse auftritt.
Die entsprechende Programmsequenz ist wie folgt:
REGIST(2) AXIS(master)
MOVELINK(dst,sync_dst,sync_beschl,sync_verz,master,1) AXIS(slave)
Weitere Informationen zum Befehl **MOVELINK** und dem Argument **Synchronisierungs_option** finden Sie in Abschnitt 3.2.180.

Abb.40

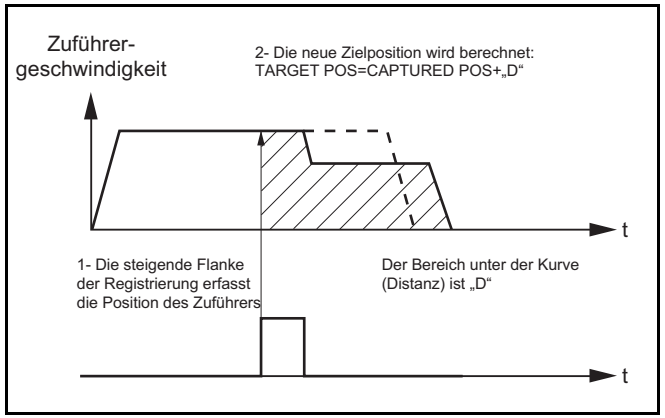
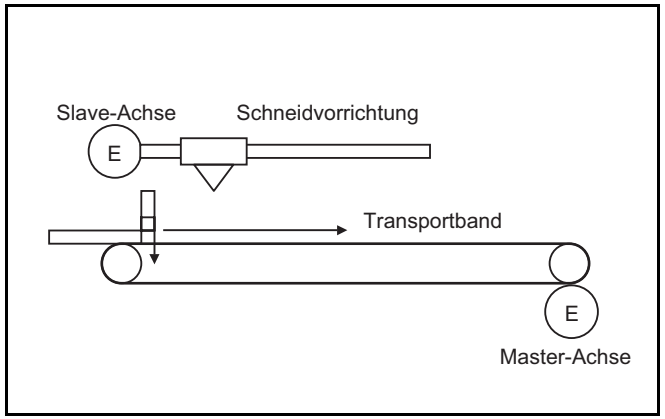
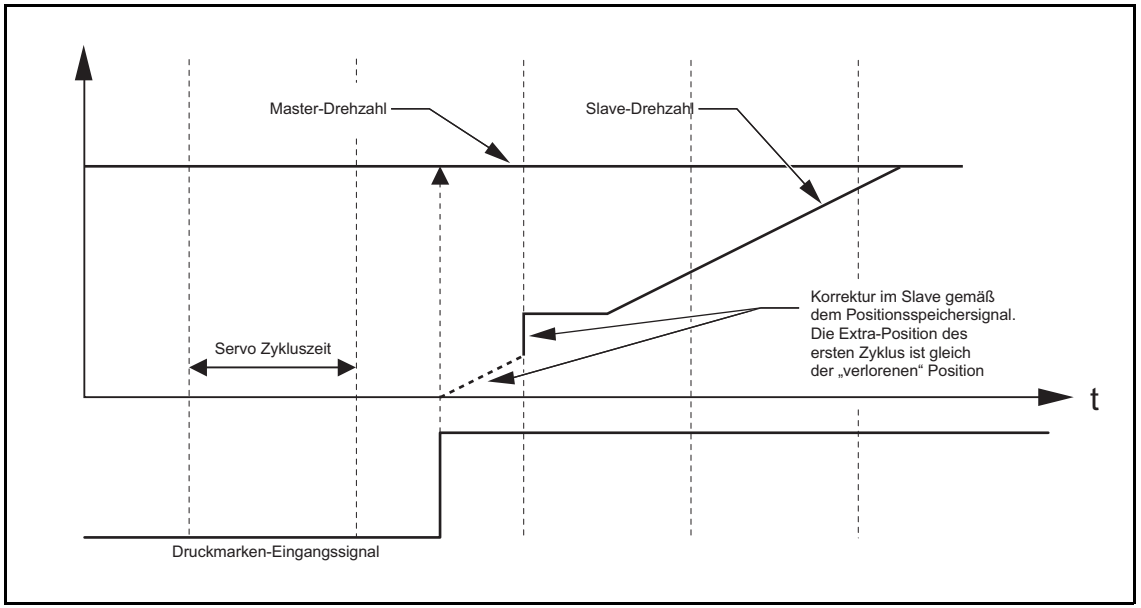


Abb.41



Die Abbildung zeigt, wie die Position der Slave-Achse mit dem Registrierungsereignis der Master-Achse korrigiert wird, um die Bewegung der Slave-Achse zu starten. Die Beeinflussung durch **SERVO_PERIOD** und die Tatsache, dass das Registrierungsereignis jederzeit innerhalb des Zeitraums von **SERVO_PERIOD** auftreten kann, wird völlig vermieden.

Abb.42



6.1.7 Datenaufzeichnung und Überwachung

Oszilloskop-Funktion in Trajexia Tools

Das Software-Oszilloskop ist standardmäßiger Bestandteil von Trajexia Tools. Das Oszilloskop kann zur Aufzeichnung und grafischen Darstellung von Achsen- und Systemparametern verwendet werden. Dies kann bei der Entwicklung und Inbetriebnahme sowie bei der Fehlersuche im Motion Control-System hilfreich sein. Weitere Informationen über das Software-Oszilloskop, seine Funktionen und Eigenschaften finden Sie in Abschnitt 5.5.4. Es gibt zwei Möglichkeiten, die Datenaufzeichnung des Oszilloskops von bestimmten Achsen- und Systemparametern auszulösen: manuell oder durch ein Programm. Die manuelle Auslösung erfolgt über das Oszilloskop-Tool. Die Parameter sind im Tabellenspeicher des Controllers gespeichert. Der Bereich des Tabellenspeichers, in dem die Parameter gespeichert sind, kann über das Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration, siehe Abschnitt 5.5.4) eingestellt werden. Bei der manuellen Auslösung kann der Benutzer

die Änderungen von Achsen- und Systemparametern in Echtzeit sehen, während das System läuft. Eine Änderung von Parameterwerten wird grafisch dargestellt, sobald die Änderung auftritt. Der Nachteil der manuellen Auslösung besteht darin, dass eine Interaktion mit dem Benutzer erforderlich ist, d. h., dass der Start der Datenaufzeichnung nicht mit der Bewegung synchronisiert ist, die analysiert wird. Außerdem ist bei der manuellen Auslösung der Aufzeichnungsbereich auf 200 Abtastungen pro Kanal begrenzt.

Verwendung des Oszilloskops

Die Alternative, Auslösung durch ein Programm, hat nicht die Beschränkungen der manuellen Auslösung der Datenaufzeichnung. Bei der Auslösung durch ein Programm werden die Achsen- und Systemparameter im Speicher des TJ1-MC__ abgelegt. Später werden die Parameter an das Oszilloskop zur grafischen Darstellung übertragen. Die Achsen- und Systemparameter sind im Tabellenspeicher des Controllers gespeichert. Der verwendete Speicherbereich ist durch die Parameter des Befehls **SCOPE** definiert. Wenn die Parameter im Tabellenspeicher abgelegt sind, kann das Oszilloskop so konfiguriert werden, dass an Stelle von Achsen- und Systemparametern ein Bereich von Tabellen-Speicheradressen angezeigt wird. Der genaue Moment, in dem die Datenaufzeichnung gestartet wird, kann exakt bestimmt werden, weil er durch den Befehl **TRIGGER** vorgegeben wird. Das heißt, der Beginn der Datenaufzeichnung wird mit der Bewegung synchronisiert. Es gibt keine Beschränkung von 200 Abtastungen pro Kanal, das Oszilloskop zeigt so viele Abtastungen (Tabelleneinträge), wie im Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration) konfiguriert sind.

Beispiel

In diesem Abschnitt wird ein praktisches Beispiel zur Verwendung der Befehle **SCOPE** und **TRIGGER** gegeben und gezeigt, wie sie in Verbindung mit dem Oszilloskop zur Überwachung von Achsenparametern und zur Fehlersuche im System verwendet werden. Weitere Informationen zu den Befehlen **SCOPE** und **TRIGGER** finden Sie in den Abschnitten 3.2.237 und 3.2.265.

Nehmen wir an, das Motion Control-System besteht aus zwei Achsen, **AXIS(0)** und **AXIS(1)**. **AXIS(0)** ist die Master-Achse. Sie führt eine einfache Vorwärtsbewegung aus. **AXIS(1)** ist die Slave-Achse. Sie muss der Master-Achse gemäß der Kosinus-Regel folgen:

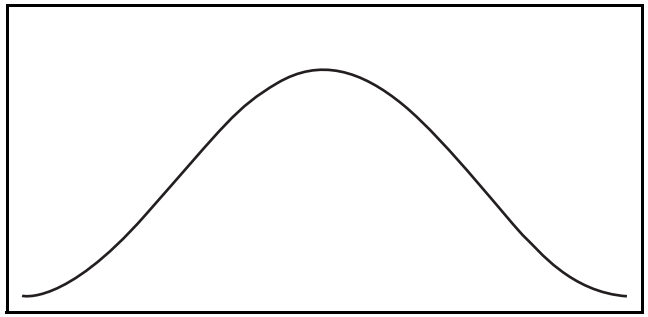
$$x_1 = end_pos \cdot \frac{1}{2}\left(1 - \cos\left(\frac{2\pi \cdot x_0}{999}\right)\right)$$

Hierbei ist x_0 die Position der Master-Achse **AXIS(0)**, und x_1 ist die Position der Slave-Achse **AXIS(1)**. Sie können die beiden Achsen mit dem Befehl **CAMBOX** synchronisieren. Weitere Informationen finden Sie in Abschnitt 3.2.42. Nehmen wir weiterhin an, der Parameter **end_pos** ist nicht konstant, sondern kann sich auf Grund von unterschiedlichen Bedingungen im Motion Control-System ändern. Der Programmteil, der die CAM-Tabelle erzeugt, ist folgender:

```
'Erste CAM-Werte
VR(end_pos)=15
aktuelle_End_pos=VR(end_pos)
FOR i=0 TO 999
    TABLE(i, VR(end_pos)*(1-COS(2*PI*i/999))/2)
NEXT i
...
schleife:
IF VR(end_pos)<>aktuelle_end_pos THEN
'CAM-Tabelle neu berechnen
    FOR i=0 TO 999
        TABLE(i, VR(end_pos)*(1-COS(2*PI*i/999))/2)
    NEXT i
    aktuelle_end_pos=VR(end_pos)
ENDIF
...
GOTO schleife
```

Der Wert **VR(end_pos)** kann von anderen Programmen oder extern von einem Controller über FINS-Messaging geändert werden. In diesem Fall muss die **CAM** -Tabelle neu berechnet werden. Die Erstellung der CAM-Tabelle ist abgeschlossen. Die Initialisierung der gewünschten Achsen- und Systemparameter für die Datenaufzeichnung ist wie folgt:

Abb.43

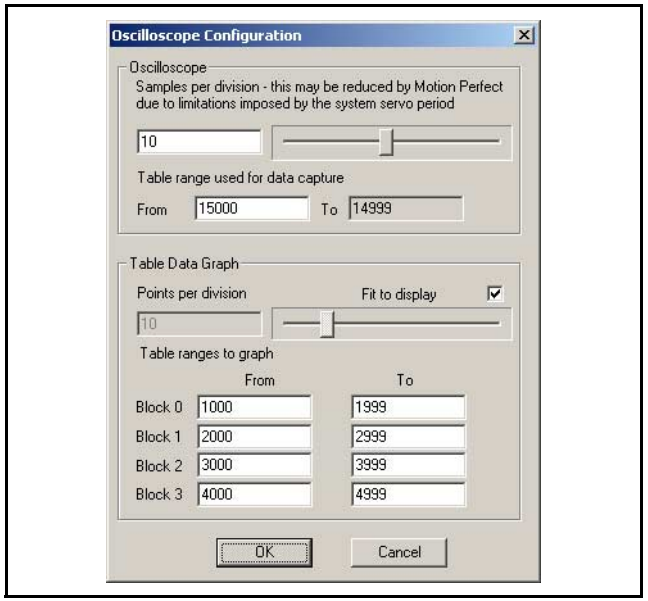


```
'Initialisierung
FOR i=0 TO 1
    BASE(i)
    ATYPE = 40
    UNITS=8192
    REP_DIST=20
    REP_OPTION=1
    FE_LIMIT=1
    DRIVE_CONTROL=11
    SPEED=8
    ACCEL=50
    DECEL=50
    DEFPOS(0)
    SERVO=ON
    CANCEL
NEXT i
WDOG=ON
BASE(1)
'Oszilloskop-Einstellungen:
'1 Abtastung alle 2 Servozyklen
'Datenspeicherung in TABLE(1000) bis TABLE(4999)
'Weil 4 Kanäle erfasst werden, ergeben sich 1000 Abtastungen pro Kanal.
'MPOS AXIS(0) ist in TABLE(1000) bis TABLE(1999) gespeichert
'DPOS AXIS(1) ist in TABLE(2000) bis TABLE(2999) gespeichert
'Der Drehmomentsollwert für AXIS(1) ist in
'TABLE(3000) bis TABLE(3999) gespeichert
'MSPEED AXIS(1) ist in TABLE(4000) bis TABLE(4999) gespeichert
'Die Erfassung beträgt 1000 Abtastungen * 2ms / Abtastung = 2 Sekunden
SCOPE(ON,2,1000,4999,MPOS AXIS(0),DPOS,DRIVE_MONITOR,MSPEED)
FORWARD AXIS(0) 'Master-Achse vorwaerts bewegen
TRIGGER 'Datenaufzeichnung und Speicherung von Parametern starten
WHILE NOT MOTION_ERROR
    'Cambox startet bei AXIS(0) Position 1
    CAMBOX(0,999,UNITS,10,0,2,1)
    WAIT UNTIL MPOS AXIS(0)<1
    'Die Erfassung beginnt, wenn sich die Master-Achse
    'an einer Position zwischen 0 and 1 befindet. Weitere Bedingungen
    'sind:
    '- Die vorherige Erfassung ist beendet
    '(SCOPE_POS=1000)
```

```
    '- Wir haben die Freigabe
(VR(trigger_aktivieren)=ON)
    IF SCOPE_POS=1000 AND VR(trigger_aktivieren)=ON THEN
        TRIGGER
        PRINT "Ausgeloest"
    ENDIF
    WAIT IDLE
WEND
HALT
```

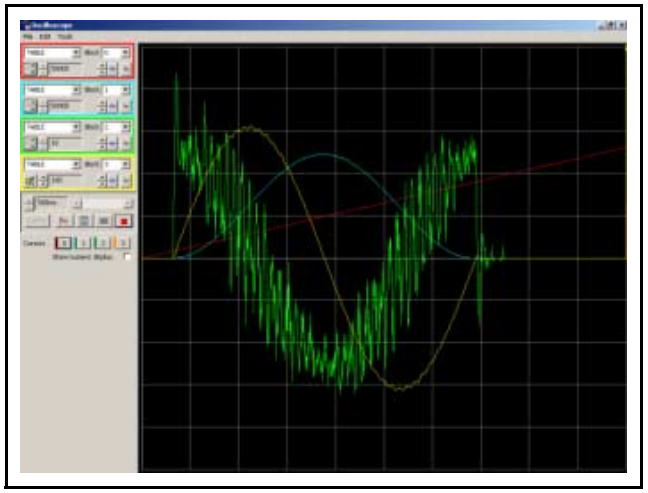
Um die Ergebnisse der Datenerfassung im Oszilloskop von Trajexia Tools betrachten zu können, müssen Sie die Einstellungen im Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration) wie in der Abbildung gezeigt vornehmen.
Außerdem müssen Sie die weitere Datenerfassung deaktivieren, um zu vermeiden, dass die Ergebnisse von zwei verschiedenen Erfassungen in den gleichen Einträgen im Tabellenspeicher vermischt werden.

Abb.44



Das Ergebnis der Datenerfassung ist in der Abbildung gezeigt.

Abb.45



Im obigen Beispiel werden mit dem Wert für den Parameter **UNITS** die Drehgeberimpulse eingestellt. Die Position der Master-Achse **MPOS AXIS(0)** ist rot dargestellt (Tabellenblock 0, Tabelle(1000) bis Tabelle(1999), siehe auch die Einstellungen im Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration)).
Die Position steigt linear an, weil die Drehzahl der Master-Achse konstant ist.
Die Sollposition der Slave-Achse **DPOS AXIS(1)** ist blau dargestellt (Tabellenblock 1, Tabelle(2000) bis Tabelle(2999), siehe auch die Einstellungen im Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration)). Dieser Graph ist eine Kosinus-Kurve. Sie entspricht der erstellten CAM-Tabelle.
Die gemessene Drehzahl der Slave-Achse MSPEED AXIS(1) ist gelb dargestellt (Tabellenblock 3, Tabelle(4000) bis Tabelle(4999), siehe auch die Einstellungen im Fenster **Oscilloscope Configuration** (Oszilloskop-Konfiguration)). Dieser Graph ist eine Sinuskurve, weil die Drehzahl eine Ableitung der Position und die Ableitung des Kosinus der Sinus ist.
Bei hohen Drehzahlen tritt etwas Welligkeit auf.
Der grüne Graph stellt das Drehmoment des Motors für die Slave-Achse dar, das mit **DRIVE_COMMAND=11** als Prozentsatz des Nenndrehmoments eingestellt ist.
Das Drehmoment ist proportional zur Beschleunigung.
Weil die Beschleunigung eine Ableitung der Drehzahl ist und die Drehzahl die Form einer Sinuskurve hat, ist die Beschleunigung (und auch das Drehmoment) eine Kosinuskurve.
Es gibt einen Spitzenwert beim Starten und einen weiteren beim Stoppen, weil es eine Diskontinuität in der Beschleunigung gibt. Außerdem gibt es eine hochfrequente Oszillation in der Drehmomentkurve, die auf eine Resonanzfrequenz hinweist, die mit den Sperrfilter-Einstellungen im Sigma-II-Servotreiber vermieden werden kann. Die hohe Frequenz wird verstärkt, weil sie auch in der Drehzahlkurve reflektiert wird. Weitere Informationen zu Sperrfilter-Einstellungen finden Sie im Sigma-II-Servotreiber-Handbuch.

Fehlersuche mit dem Oszilloskop

Wenn die gewünschten Daten erfasst und im Tabellenspeicher aufgezeichnet sind, können Sie das Oszilloskop zur Visualisierung dieser Daten verwenden. Dies kann bei der Inbetriebnahme und bei der Fehlersuche im System hilfreich sein. Dieser Abschnitt gibt ein Beispiel dafür, wie ein Fehler, der schwierig zu analysieren ist, mit den erfassten Daten und dem Oszilloskop deutlich erklärt und behoben werden kann.

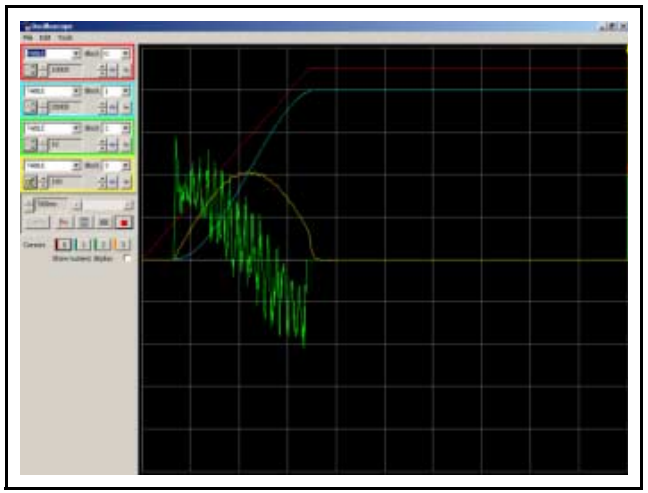
Der Parameter **end_pos**, der die Werte in der CAM-Tabelle definiert, ist von äußeren Bedingungen des Systems abhängig. Deshalb kann er durch ein Programm verändert werden, das in einer anderen Task oder sogar in einem Steuergerät läuft, das über FINS-Kommunikation angeschlossen ist, während das Hauptprogramm läuft, das zwei Achsen miteinander synchronisiert. Nehmen wir an, dass diese Bedingungsänderungen, die eine Änderung des Parameters **end_pos** zur Folge haben, meist dann passieren, wenn die Achsen nicht synchronisiert sind, d. h., wenn der Befehl **CAMBOX** nicht ausgeführt wird. Nehmen wir weiterhin an, dass sich die Bedingung sehr selten dann ändert, wenn die Achsen synchronisiert sind. Die Änderung des Parameters **end_pos** löst eine Neuberechnung der CAM-Tabelle aus, während der Befehl **CAMBOX** ausgeführt wird. Dies hat zur Folge, dass der Teil der Sollposition der Slave-Achse dem Profil vor der Änderung folgt, der andere Teil jedoch dem Profil nach der Änderung folgt. Schließlich führt dies zu einem Abbruch des Profils, was eine unbestimmte Drehzahl der Achse zur Folge hat und folgenden Fehler verursacht: WDOG schaltet ab, und alle Achsen werden gestoppt.
Das obige Szenario ist schwierig zu analysieren, wenn Sie nicht wissen, was passiert. Der Benutzer sieht nur, dass die Slave-Achse alle paar Stunden oder noch seltener einen Fehler hat. Aber das Oszilloskop kann deutlich zeigen, wo das Problem liegt. Um das Oszilloskop einsetzen zu können, müssen alle gewünschten Parameter zum Zeitpunkt eines Fehlers erfasst sein. Dies kann dadurch erreicht werden, dass die Anwendungsprogramme auf eine bestimmte Weise arrangiert werden. Eine gute Programmiertechnik besteht darin, ein separates Startprogramm einzurichten, das automatisch beim Hochfahren des Systems gestartet wird, die Integrität des Systems überprüft und kontrolliert, ob alle erwarteten

Geräte angeschlossen und initialisiert sind. Ein Beispiel für ein Startprogramm finden Sie in Abschnitt 6.1.1. Es wird empfohlen, das Startprogramm nach seiner Ausführung nur ein Programm starten zu lassen, das die Sicherheit und Integrität der Anwendung sowie die Ausführung aller anderen Anwendungsprogramme überwacht. Dieses Programm wird normalerweise als SHELL-Programm bezeichnet. Weitere Informationen zur Entwicklung eines SHELL-Programms finden Sie in Abschnitt 6.2.1.
Nehmen wir an, dieses Programm ist so aufgebaut, dass es folgenden Code enthält:

```
'Wenn ein Fehler auftritt, werden alle Programme
gestoppt. Es werden
'keine neuen Oszilloskop-Aufzeichnungen vorgenommen.
Außerdem ist in den
'ausgewaehlten TABLES die letzte Datenaufzeichnung
gespeichert, bei der
'der Fehler aufgetreten ist. Daher koennen wir diese
Aufzeichnung
'zurueckverfolgen und sie analysieren.
schleife:
    IF MOTION_ERROR<>0 THEN HALT
GOTO schleife
```

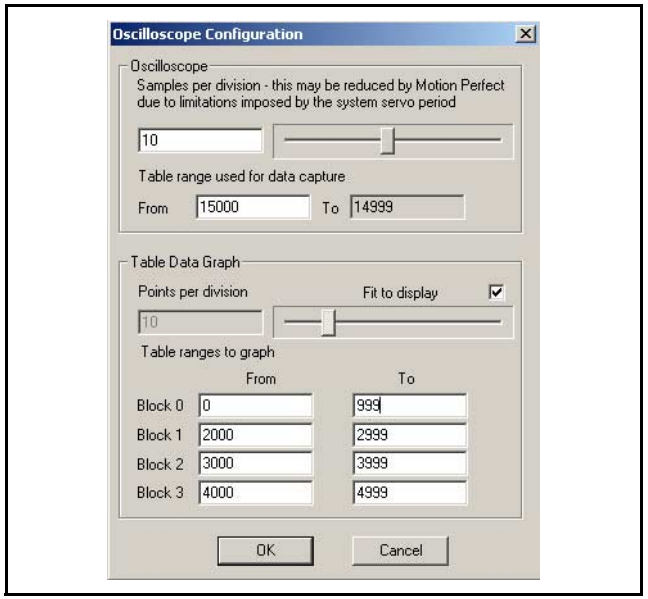
Dieser Programmiercode bewirkt, dass alle Programme und die Aufzeichnung gestoppt werden, wenn bei einer Achse ein Fehler auftritt. Die Daten sind bereits im Tabellenspeicher erfasst; wir können das Oszilloskop dazu verwenden, den Status der gewünschten Parameter in dem Moment zu sehen, als der Fehler auftrat. Mit den Oszilloskop-Einstellungen wie in Abb.44 ist das Ergebnis für das oben beschriebene Szenario in Abb.46 dargestellt. Die gemessene Position der Master-Achse, die rot dargestellt ist, scheint nicht die Ursache zu sein, weil dort keine Diskontinuität aufgetreten ist. Ein mechanisches Problem können wir ebenfalls ausschließen, weil das Drehmoment, das grün dargestellt ist, niedrige Werte hat. Zum Zeitpunkt des Auftretens war die Drehzahl der Slave-Achse, die gelb dargestellt ist, gleichmäßig und niedrig, daher gibt es auch hier kein Problem.

Abb.46



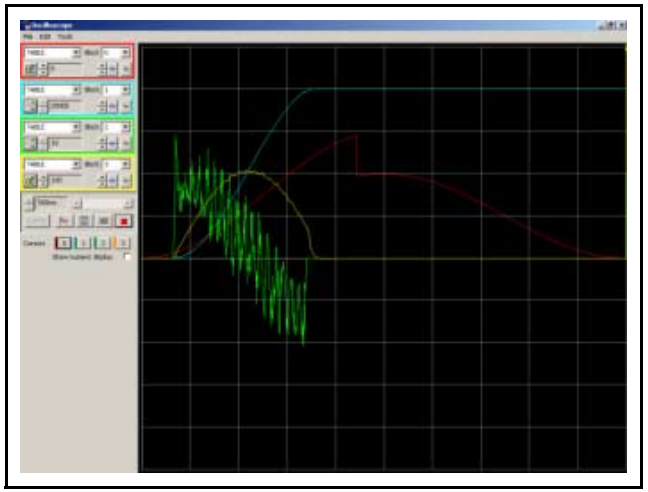
Der nächste Schritt besteht darin, die CAM-Tabelle zu analysieren, um zu sehen, welche Werte für die Slave-Achse zum Erreichen der Position verwendet wurden. Zu diesem Zweck ändern wir die Oszilloskop-Konfiguration so, dass ein Werteblock von Table(0) bis Table(999) in rot angezeigt wird, weil dies die Tabellenspeicher-Einträge sind, bei denen die CAM-Tabelle angelegt wurde (siehe den Programmteil oben, mit dem die CAM-Tabelle angelegt wird). Die geänderte Konfiguration ist in der Abbildung gezeigt.

Abb.47



Das Ergebnis ist in der Abbildung dargestellt. Die rote Kurve zeigt deutlich eine Diskontinuität in den Positionswerten, denen die Slave-Achse folgen muss. Weil die Drehzahl eine Ableitung der Position ist, bekommt die Drehzahl an dem Punkt der Diskontinuität der Positionskurve einen hohen Wert. (Theoretisch ist dieser Wert gleich unendlich, in der Praxis ist er einfach sehr hoch). Dadurch wird der Fehler verursacht. Die rote Kurve zeigt, wo die Ursache des Problems liegt. Die Amplitude der Kosinuskurve und damit der Parameter **end_pos** ist während der Ausführung des Befehls **CAMBOX** geändert worden. Die Lösung ist einfach: Eine Änderung des Parameters **end_pos** während der Ausführung von **CAMBOX** muss verhindert werden. Dies kann dadurch erfolgen, dass die Programme in Trajexia oder in einem anderen Controller modifiziert werden (wenn der Parameter außerhalb des Umfangs der Anwendungsprogramme geändert wird, z. B. durch eine FINS-Meldung).

Abb.48



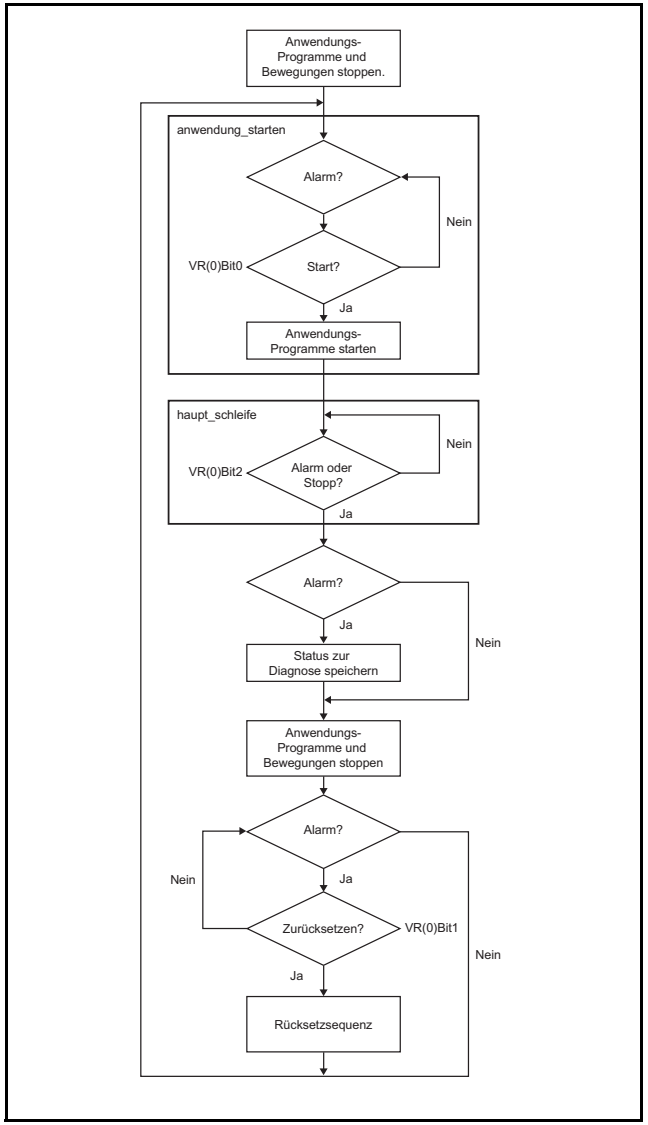
Die Zeitbasis der CAM TABLE-Punkte ist nicht die gleiche wie für die Erfassung der anderen Signale. Die Unterbrechung in CAM (rote Kurve) fällt mit dem Zeitpunkt der Unterbrechung der Bewegung zusammen. Um dies zu analysieren, überprüfen Sie die Positionswerte einzeln mit einem Tabellenkalkulationsprogramm. Um die Punktwerte im Einzelnen zu analysieren, können Sie die TABLE-Punkte für eine komplexere Analyse in ein Tabellenkalkulationsprogramm exportieren.

6.2 Praktische Beispiele

6.2.1 Shell-Programm

Eine gute Programmiertechnik erfordert ein gutes Shell-Programm Ein Shell-Programm startet und stoppt die Anwendungsprogramme und setzt diese zurück. Das Shell-Programm ist nicht notwendig, gibt den Anwendungen aber Struktur und macht die Methode zur Programmierung des Motion-Controllers effektiver.
Ein Beispiel eines Shell-Programms ist unten gezeigt. Achten Sie darauf, das Programm an die spezifischen Bedürfnisse der Anwendung anzupassen. Überprüfen Sie die korrekte Ausführung, bevor Sie sich auf den sicheren Betrieb des Programms verlassen. Dieses Programm wird normalerweise so eingestellt, dass es beim Hochfahren mit niedriger Priorität ausgeführt wird.

Abb.49



Beispiel

```
'================================================
'Beispiel eines Shell-Programms
'DIESE VERSION WURDE FUER MECHATROLINK-SERVOS ENTWICKELT
'Passen Sie dieses Programm gemAESS Ihrer Anwendung an
'================================================
'in diesem Beispiel werden bei einem Fehler alle
'PROGRAMME UND BEWEGUNGEN GESTOPPT
'Wir verwenden eine globale Variable (VR(0)) fUEr das
Programm-
'MANAGEMENT.
' VR(0)Bit0 zum Starten der Anwendung (steigende Flanke)
' VR(0)Bit1 zum RESET aller Alarme (steigende Flanke)
' VR(0)Bit2 zum Stoppen der Anwendung (steigende Flanke)
'================================================
'Dieses Beispiel gilt für eine Anwendung mit drei
'Servos
'AXIS 1, 2 und 3
'================================================

'Initialisierung von Variablen
max_achse=2
'Projekt_status
```

```
'=0 Anfangswert
'=1 Programme gestoppt, aber kein Fehler
'=2 Programme gestoppt und Fehler
'=3 Programme laufen
Projekt_status=0

'Alarm_status
'=0 Nichts
'=1 Alarm bei Achse 0
'=2 Alarm bei Achse 1
'=3 Alarm bei Achse 3
'=4 Alarm bei MECHATROLINK
'=5 Intakt
Alarm_status=5

'Aktion
'=0 Nichts
'=1 Für Neustart Reset druecken
'=2 Ruecksetzung
'=3 OK
Aktion=3

GOSUB alle_stoppen
GOSUB anwendung_starten

schleife:
    'Wenn Fehler oder Stopp-Befehl
    IF MOTION_ERROR<>0 OR READ_BIT(1,0) THEN GOSUB alarm_sequenz

    'Servotreiber-Warnung loeschen, falls noetig
    IF (DRIVE_STATUS AXIS(0) AND 2)>0 THEN DRIVE_CLEAR AXIS(0)
    IF (DRIVE_STATUS AXIS(1) AND 2)>0 THEN DRIVE_CLEAR AXIS(1)
    IF (DRIVE_STATUS AXIS(1) AND 2)>0 THEN DRIVE_CLEAR AXIS(1)
GOTO schleife

alarm_sequenz:
    IF MOTION_ERROR<>0 THEN
        projekt_status=2
        aktion=1
        'FEHLERDIAGNOSE
```

```
        'Ueberpruefung auf MECHATROLINK-Fehler
        IF (AXISSTATUS AXIS(0) AND 4)<>0 THEN
            alarm_status=4
        ELSEIF (AXISSTATUS AXIS(1) AND 4)<>0 THEN
            alarm_status=4
        ELSEIF (AXISSTATUS AXIS(1) AND 4)<>0 THEN
            alarm_status=4
        ELSE
            'Ueberpruefung auf Achsenfehler
            alarm_status=ERROR_AXIS+1
        ENDIF
    ELSE
        projekt_status=1
    ENDIF
    GOSUB alle_stoppen
    GOSUB alle_zuruecksetzen
    GOSUB anwendung_starten
RETURN

alle_stoppen:
    'STOPPEN DER PROGRAMME
    STOP "ANWENDUNG"

    'STOPPEN DER ACHSEN
    FOR i= 0 TO max_achse
        BASE(i)
        CANCEL(1) 'Abbruch NTYPE
        WA(1)
        CANCEL(1) ' moeglichen Programm-Befehlsspeicher löschen
        CANCEL 'Abbruch MTYPE
    NEXT i
    'RUN-Befehl in den Servos freigeben
    WDOG=0
    'Positionsregelkreis oeffnen
    FOR i= 0 TO max_Achse
        BASE(i)
        WAIT IDLE
        SERVO=0
    NEXT i
RETURN
```

```
alle_zuruecksetzen:
    WHILE MOTION_ERROR<>0
        'Auf steigende Flanke in RESET-Eingang warten
        WAIT UNTIL READ_BIT(2,0)=0
        WAIT UNTIL READ_BIT(2,0)=1
        aktion=2
        'Fehlerhafte Servos zuruecksetzen
        FOR i=0 TO max_achse
            BASE(i)
            'Im Fall von ML-II-Fehler ist die Ruecksetzung:
            IF (AXISSTATUS AND 4)<>0 THEN
                MECHATROLINK(0,0)
                WA(3000)
                DATUM(0)
                RUN "startup",1
                STOP
            ENDIF
            'Im Fall eines Servotreiber-Fehlers
            IF (AXISSTATUS AND 8)<>0 THEN DRIVE_CLEAR
        NEXT i
        WA(100)
        'Im Fall eines Achsenfehlers
        DATUM(0)
    WEND
    projekt_status=1 'Gestoppt, aber kein Fehler
    alarm_status=5
    aktion=3
RETURN

anwendung_starten:
    'Auf steigende Flanke in Bit 0 von VR(0) warten
    WHILE READ_BIT(0,0)=0
        IF MOTION_ERROR<>0 THEN RETURN
    WEND
    WHILE READ_BIT(0,0)=1
        IF MOTION_ERROR<>0 THEN RETURN
    WEND
    RUN "ANWENDUNG"
    projekt_status=3 'Anwendung laeuft
RETURN
```

6.2.2 Initialisierungsprogramm

Das Initialisierungsprogramm setzt die Parameter für die Achsen. Diese Parameter sind abhängig von der Auflösung des Motor-Drehgebers und der maximalen Motordrehzahl.

Hinweis:
Diese Informationen finden Sie im Datenblatt für den Servotreiber und den Motor.

```
'================================================
'Beispiel eines INITIALISIERUNGSProgramms
'DIESE VERSION WURDE FUER MECHATROLINK-SERVOS ENTWICKELT
'Passen Sie dieses Programm gemaeSS Ihrer Anwendung an
'================================================
BASE(x)
Neustart=0
traegheits_verhaeltnis=eingestelltes_last_traegheits_v
erhaeltnis

'----------------------------
'Beispiel 1
'Motordaten SGMAH-01AAA61D-OY
'----------------------------
drehgeber_aufloesung=2^13 '13-Bit Drehgeber
max_drehzahl=5000 '5000 rpm max. Drehzahl

'----------------------------
'Beispiel 2
'Motordaten SGMAH-01A1A61D-OY
'----------------------------
drehgeber_aufloesung=2^16 '16-Bit Drehgeber
max_drehzahl=5000 '5000 rpm max. Drehzahl

'----------------------------
'Parameter in den Servo SCHREIBEN
'----------------------------
DRIVE_WRITE($103,2,traegheits_verhaeltnis)
'Traegheitsverhaeltnis schreiben
DRIVE_READ($110,2,10)
```

```
IF VR(10)<>$0012 THEN
    DRIVE_WRITE($110,2,$0012,1)
    'Pn110=0012h (Autotuning deaktiviert)
    neustart=1
ENDIF
DRIVE_READ($202,2,10)
IF VR(10)<>1 THEN
    DRIVE_WRITE($202,2,1,1)
    'Pn202=1 (Uebersetzungsverhaeltnis-Zaehler im Antrieb. Die Standardeinstellung ist 4)
    neustart=1
ENDIF
DRIVE_READ($511,2,10)
IF VR(10)<>$6548 THEN
    DRIVE_WRITE($511,2,$6548,1)
    'Pn511 Druckmarkeneingaenge im Servotreiber einstellen
    neustart=1
ENDIF
DRIVE_READ($81E,2,10)
IF VR(10)<>$4321 THEN
    DRIVE_WRITE($81E,2,$4321,1)
    'Pn81E=$4321 Um die Digitaleingaenge im Servotreiber
    'fuer das Lesen durch das Wort DRIVE_INPUTS verfuegbar zu machen
    neustart=1
ENDIF
IF neustart=1 THEN DRIVE_RESET

'----------------------------
'Anfangsverstaerkung fuer MECHATROLINK_SPEED
'----------------------------
'Erfahrungsgemaess ist diese Einstellung ein guter Startpunkt
P_GAIN=INT(214748.3648*max_drehzahl/drehgeber_aufloesung)
'Dies ist der optimale Wert. Noetigenfalls einstellen
VFF_GAIN=INT(60000*1073741824/drehgeber_aufloesung/max_drehzahl)

'----------------------------
'Anfangsverstaerkung für MECHATROLINK_POSITION-Betriebsart
'----------------------------
'Veraendern Sie die Steifigkeit (Fn001) gemaess dem 'mechanischen System
'Aendern Sie noetigenfalls die Verstaerkung der Drehzahlvorsteuerung Pn109
```

```
'----------------------------
'Anfangsparameter der AXIS
'----------------------------
'Wenn auf 1 gesetzt (und Pn202=Pn203=1), sind die UNITS 'Drehgeberimpulse
UNITS=1
'Theoretischer Schleppfehler laesst den Motor bei "max_drehzahl" laufen
'ohne VFF_GAIN in MECHATROLINK SPEED
FE_LIMIT=1073741824/P_GAIN/UNITS
'SPEED ist auf 1/3 "max_drehzahl eingestellt
SPEED=(max_drehzahl73)*drehgeber_aufloesung/60/UNITS
'ACCEL in 200 ms von 0 auf "max_Drehzahl"
ACCEL=SPEED/0,2
'DECEL in 200 ms von "max_drehzahl" auf 0
DECEL=SPEED/0,2
```

6.2.3 Programm für eine Achse

Dieses Programm ist ein einfaches Programm für den Betrieb mit einer einzelnen Achse.

Abb.50

Beispiel

```
'GOSUB nullpunktsuch
BASE(0)
DEFPOS(0)
WA(100)
schleife:
    MOVE(1440)
    WAIT IDLE
    WA(100)
GOTO schleife
```

Die Einheit in diesem Beispiel ist Grad, deshalb:

- 13-Bit-Drehgeber
- Pn202=32
- Pn203=45
- **UNITS=32**

Der Graph in der Abbildung ist typisch für diese Punkt-zu-Punkt-Bewegung mit linearer Beschleunigung). Bitte beachten Sie Folgendes:

- Während der linearen Beschleunigung ist die Kurve der Position eine Parabel (weil die Drehzahl eine Ableitung der Position ist).
- Bei konstanter Drehzahl ist der Graph der Position eine Gerade.
- Bei linearer Verzögerung ist der Graph der Position eine umgekehrte Parabel.
- Bei Stopp ist der Graph der Position konstant.
- Wenn ein Überlauf auftritt (**MPOS>=REP_DIST**), springt die Position auf 0, wenn **REP_OPTION=1** oder auf **-REP_DIST**, wenn **REP_OPTION=0**.

- Der Schleppfehler ist proportional zur Drehzahl, wenn Sie nur Proportionalverstärkung im Positionsregelkreis verwenden.
- Das Drehmoment, das durch den Parameter **DRIVE_MONITOR** als Prozentsatz des Nenndrehmoments des Motors vorgegeben wird (wenn Sie **DRIVE_CONTROL=11** einstellen), ist nach folgender Formel proportional zur Beschleunigung:

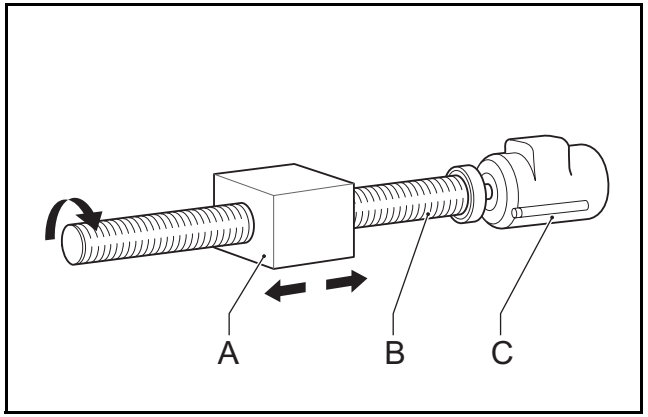
  $Drehmoment_{gesamt} = J_{gesamt} \times \alpha + Drehmoment_{Reibung}$

  Hierbei ist $Drehmoment_{Reibung}$ normalerweise klein, α ist die Winkelbeschleunigung und J die Trägheit des Systems.

6.2.4 Position mit Produkterkennung

Eine Kugelspindel bewegt sich mit Schleichganggeschwindigkeit vorwärts, bis sie ein Produkt erreicht, dann wird ein Mikroschalter (IN(2)) eingeschaltet.
Die Kugelspindel wird sofort gestoppt; die Position, an der das Produkt erfasst wurde, wird angezeigt, und die Kugelspindel kehrt mit schneller Drehzahl zur Startposition zurück.

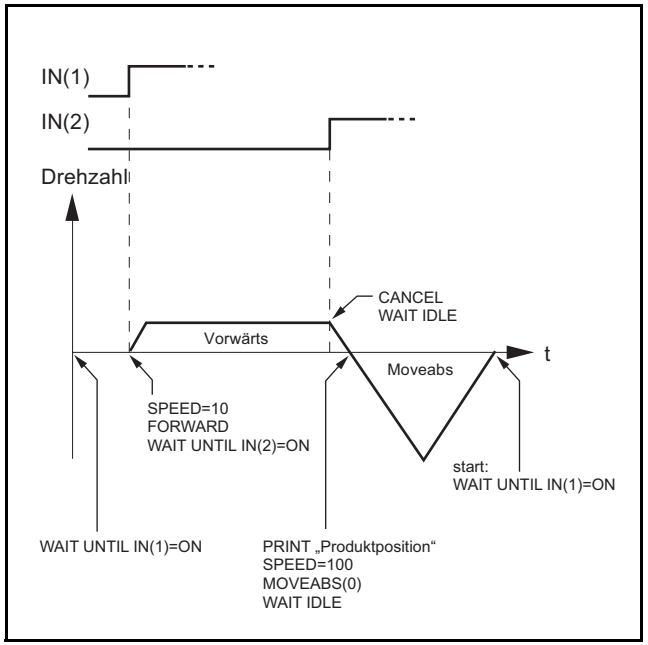
Abb.51



Beispiel

```
start:
    WAIT UNTIL IN(1)=ON
    SPEED=10
    FORWARD
    WAIT UNTIL IN(2)=ON
    prod_pos=MPOS
    CANCEL
    WAIT IDLE
    PRINT "Produktposition : "; prod_pos
    SPEED=100
    MOVEABS(0)
    WAIT IDLE
GOTO start
```

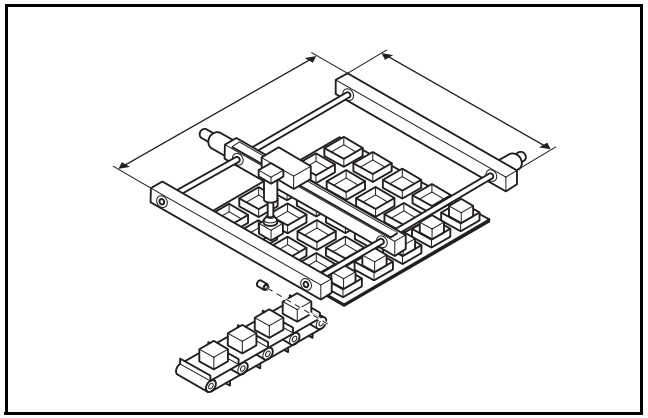
Abb.52



6.2.5 Position in einem Gitter

Die Seiten einer quadratischen Palette sind 1 m lang. Sie ist in ein 5 x 5 Gitter unterteilt. Jede der Positionen im Gitter enthält einen Karton, der mit dem gleichen Quadratmuster von 100 mm x 100 mm befüllt werden muss. Eine Verteilerdüse, die über Digitalausgang 8 gesteuert wird, muss beim Befüllen des Kartons eingeschaltet werden und ansonsten ausgeschaltet bleiben.

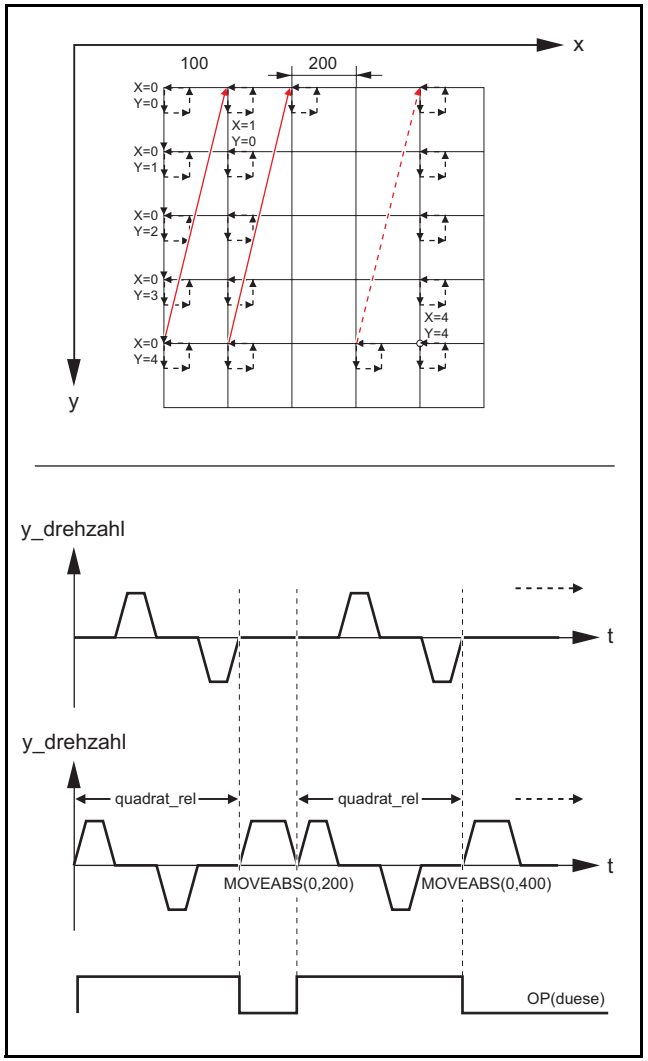
Abb.53



Beispiel

```
duese = 8
start:
    FOR x = 0 TO 4
        FOR y = 0 TO 4
            MOVEABS(x*200, y*200)
            WAIT IDLE
            OP(duese, ON)
            GOSUB quadrat_rel
            OP(duese, OFF)
        NEXT y
    NEXT x
GOTO start
quadrat_rel:
    MOVE(0, 100)
    MOVE(100, 0)
    MOVE(0, -100)
    MOVE(-100,0)
    WAIT IDLE
    WA(1000)
RETURN
```

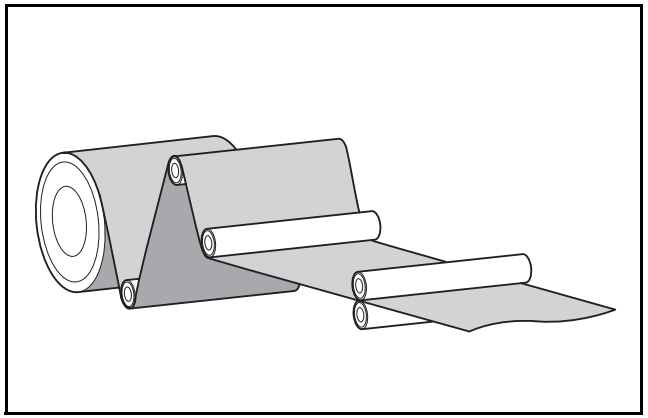
Abb.54



6.2.6 Tüten-Zuführprogramm

Eine Tüten-Zuführmaschine führt Plastikfolie über eine bestimmte Strecke zu, die vom Bediener festgelegt ist. Die Abbildung zeigt einen typischen Tüten-Zuführer, der Bestandteil der Maschine ist.

Abb.55



Tüten-Zuführmaschinen haben zwei Betriebsarten.

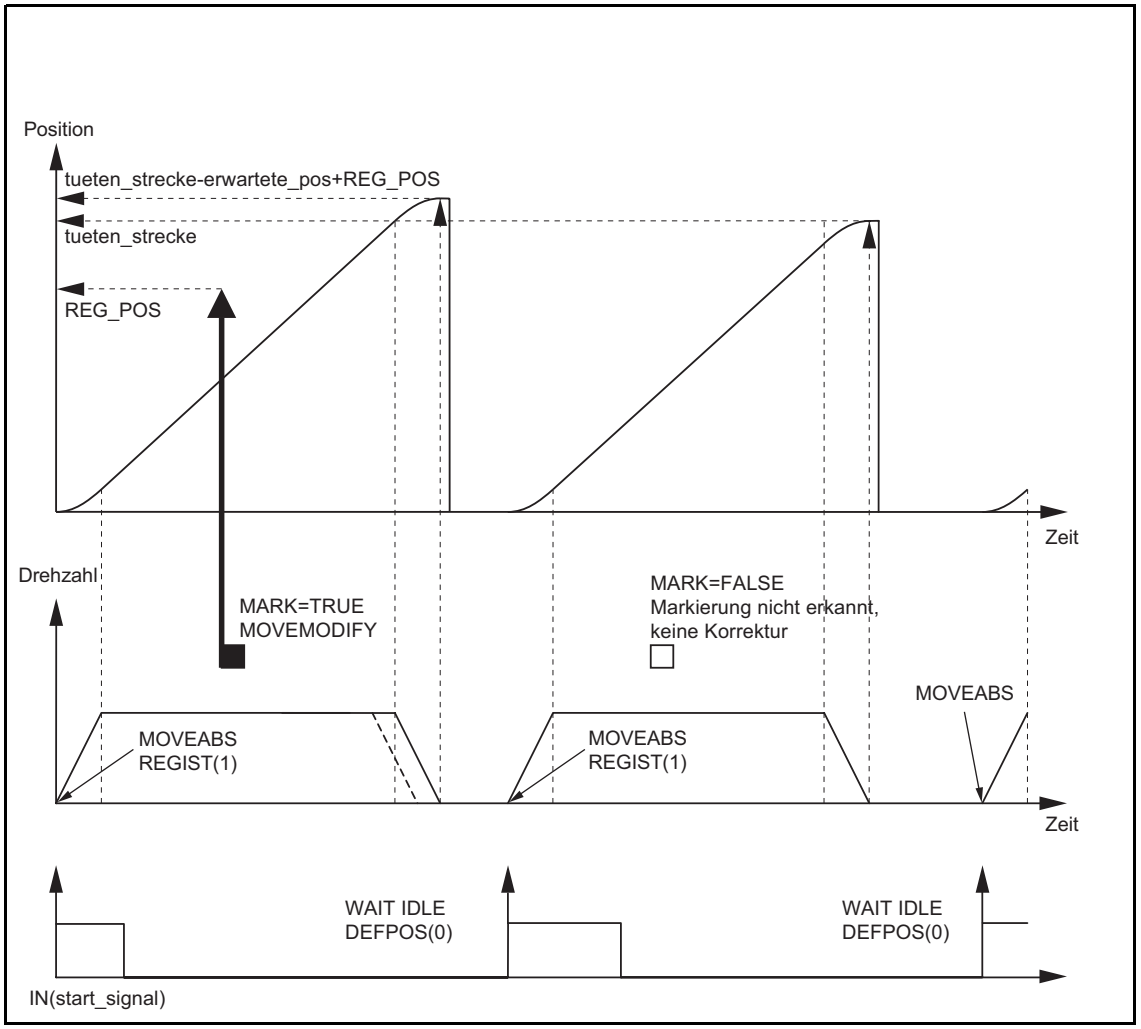
- Ohne Markierung: Führt die Folie über eine festgelegte Strecke zu, für Folien von matter Farbe
- Mit Markierung: Führt die Folie bis zu einer gedruckten Markierung auf der Folie zu.

Das Programm in diesem Abschnitt zeigt den typischen Code für eineTüten-Zuführmaschine.

Beispiel

```
'================================================
'TUETEN-ZUFUEHR-Programm
'================================================
'Wenn beim Einsatz von Markierungen eine Markierung fehlt,
'wird die theoretische Strecke zugefuehrt. Aber wenn
die Markierung bei
'mehreren Tueten hintereinander fehlt, wird der
Betrieb gestoppt.
'Ein digitaler Ausgang wird zu einem bestimmten
Zeitpunkt aktiviert, um
'die Tuete abzuschneiden.
'================================================

'Initialisierung von Variablen
start_signal=7
max_fehlschlag=3
programm_alarm=0
fehlschlag=0
zufuehr_achse=2
BASE(zufuehr_achse)
'Positionszaehler (MPOS,DPOS) reicht von 0 bis 999999
'und wieder 0
UNITS=27
SPEED=100
ACCEL=1000
DECEL=1000
REP_DIST=1000000
REP_OPTION=1
SERVO=ON
WDOG=ON

'Hauptprogramm
schleife:
    'Aktuelle Position als null definieren
    DEFPOS(0)
```

Abb.56



```
    'Auf steigende Flanke im Digitaleingang warten
    '"Start_signal"
    WAIT UNTIL IN(start_signal)=0
    WAIT UNTIL IN(start_signal)=1

    'Bewegung über Tuetenlaenge
    MOVEABS(tueten_strecke)
    WAIT UNTIL MTYPE=2 'Um zu ueberpruefen, ob MOVEABS ausgefuehrt wird

    'Wenn Markierungen eingesetzt werden, Ausloesung aktivieren
    'MARK=FALSE wenn ausgeloest und TRUE wenn nicht ausgeloest
    IF mit_markierung AND MARK THEN
        REGIST(1)
        WAIT UNTIL MARK=0
    ENDIF

    'Warten, bis Bewegung beendet oder Markierung erkannt ist
    WAIT UNTIL MTYPE=0 OR (MARK AND mit_markierung)

    'mit Markierung arbeiten
    IF mit_markierung THEN
        IF MARK THEN 'Wenn die Markierung erkannt worden ist, wird die Position korrigiert
            MOVEMODIFY(tueten_strecke-erwartete_pos+REG_POS)
            fehlschlag=0

        ELSE 'Wenn die Markierung nicht erkannt wurde
            PRINT "Markierung nicht erkannt"
            fehlschlag=fehlschlag+1
            IF fehlschlag>max_fehlschlagTHEN 'Nach mehreren Fehlerfassungen Anwendung stoppen
                PRINT "Markierung definitiv verloren"
                programm_alarm=3
                STOP
            ENDIF
        ENDIF
    ENDIF

    'Warten, bis Zufuehrbewegung beendet ist
    WAIT IDLE
GOTO schleife
```

6.2.7 CAM-Tabelle innerhalb eines Programms

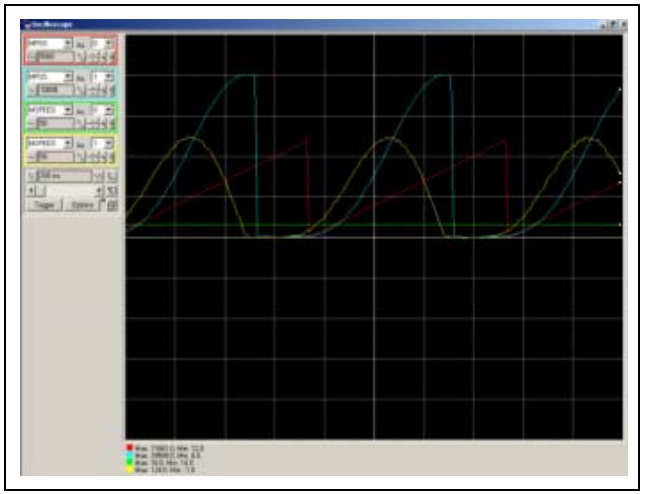
Es wird gezeigt, wie eine CAM-Tabelle innerhalb eines Programms angelegt und der Bewegungsbefehl **CAMBOX** verwendet wird. Das verwendete Profil ist das COS-Quadratprofil. Dies ist ein recht typisches Profil für Zuführ-Anwendungen, da:

- die Bewegung eine gleichmäßige Beschleunigung ohne plötzliche Beschleunigungsänderungen gewährleistet, sodass der Materialschlupf minimiert wird,
- eine schnelle Verzögerung möglich ist und die Zykluszeit reduziert wird. Während der Verzögerung gibt es keinen Materialschlupf, und die Reibung hilft bei der Verzögerung bis auf Nulldrehzahl.

Beispiel

```
start:
    GOSUB tabelle_fuellen
    WDOG=1 'Servos auf RUN einstellen
    BASE(1)
    SERVO=1 'Positionsregelkreis in Achse 1 aktivieren
    BASE(0)
    SERVO=1 'Positionsregelkreis in Achse 0 aktivieren
    'Der Positionszähler zaehlt von 0 bis 11999
    'und dann wieder zurueck auf 0
    REP_OPTION=1
    REP_DIST=12000
    SPEED=200
    FORWARD

BASE(1)
schleife:

CAMBOX(in_tbl,ende_tbl,1,sync_dst,master,opt,start)
    WAIT IDLE
GOTO schleife
```

Abb.57



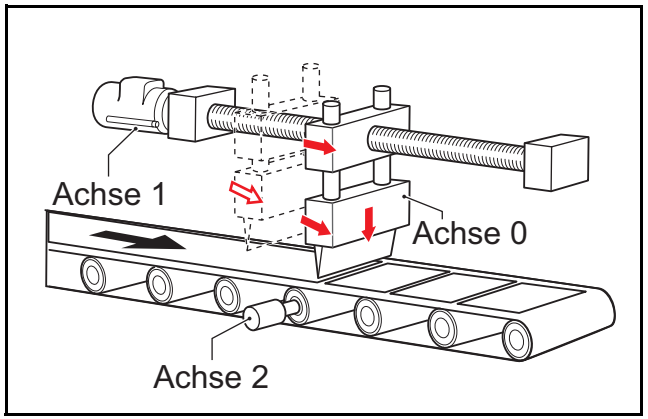
```
tabelle_fuellen:
    'Die Form von CAM ist in TABLE(0) bis
    'TABLE(360) gespeichert
    n_punkte=360
    in_tbl=0
    ende_tbl=in_tbl+n_punkte
    'Distanz des Masters, um die CAM zu erstellen
    sync_dst=10000
    'Master-Achse
    Master=0
    'Die CAM startet genau dann, wenn der Master die
    'Position "start" erreicht
    opt=2
    start=1000

    k=100
    'Die TABLE mit der entsprechenden Wellenform
fuellen
    FOR i= in_tbl TO ende_tbl
        TABLE(i,(k*(COS(PI*i/n_punkte)-1))^2)
    NEXT i
RETURN
```

6.2.8 Schneidvorrichtungsprogramm

Ein Beispiel für ein Schneidvorrichtungsprogramm.
In dieser Anwendung gibt es drei Achsen:

- Achse 0, schneid_achse, die das Schneidwerkzeug vertikal bewegt.
- Achse 1, fliegende_achse, die das Schneidwerkzeug horizontal bewegt.
- Achse 2, band_achse, die das Material transportiert.

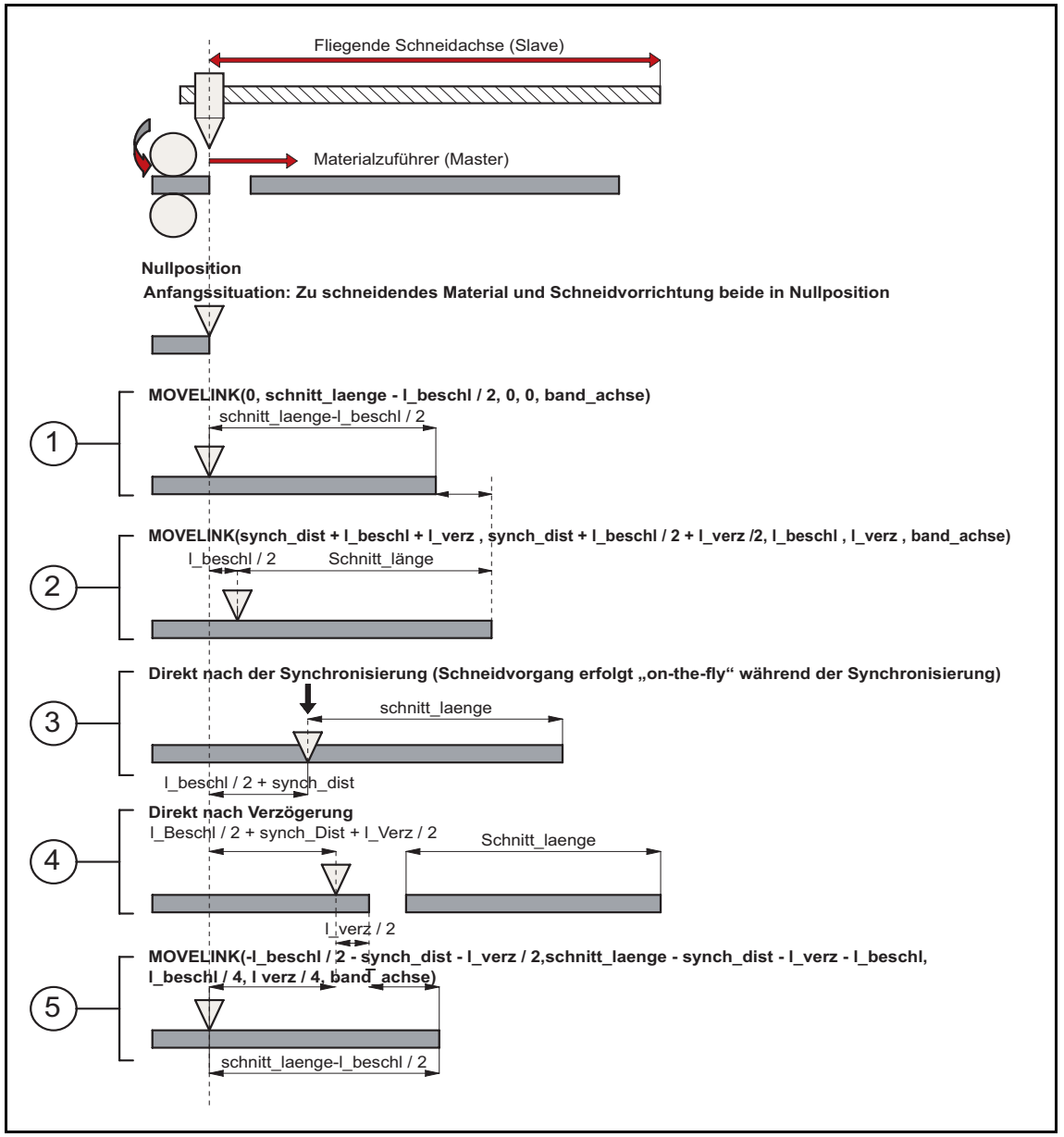
Abb.58



Beispiel

```
'================================================
'Schneidvorrichtungs-Programm
'================================================
'Typisches Beispiel für eine Schneidvorrichtungs-
Anwendung.
'Eine Achse (band_achse) transportiert das Material.
'Die zweite Achse (fliegende_achse) transportiert das
Schneidwerkzeug selbst.
'Die dritte Achse (schneid_achse) bewegt das
Schneidwerkzeug nach oben und unten.
'Die Strecke bei der Synchronisation muss lang genug sein,
'um den Schnitt bei maximaler Drehzahl zu ermoeglichen.
'Das Zurueckfahren des Schneidwerkzeugs erfolgt mit
einer solchen
'Geschwindigkeit, dass die Wartezeit null ist
(Optimierung der
'Bewegung).
'Es wird davon ausgegangen, dass alles so
'berechnet wurde, dass die maximale Motordrehzahl bei
' maximaler Bandgeschwindigkeit nicht ueberschritten wird
'================================================
schnitt_zaehler=0
band_achse=2
schneid_achse=0
fliegende_achse=1

SERVO AXIS(band_achse)=ON
SERVO AXIS(fliegende_achse)=ON
SERVO AXIS(schneid_achse)=ON
WDOG=ON

'ERSTER ZYKLUS

'Ersten Materialschnitt machen
MOVEABS(end_pos) AXIS(schneid_achse)
WAIT UNTIL MTYPE AXIS(schneid_achse)=2
WAIT IDLE AXIS(schneid_achse)
```

Abb.59



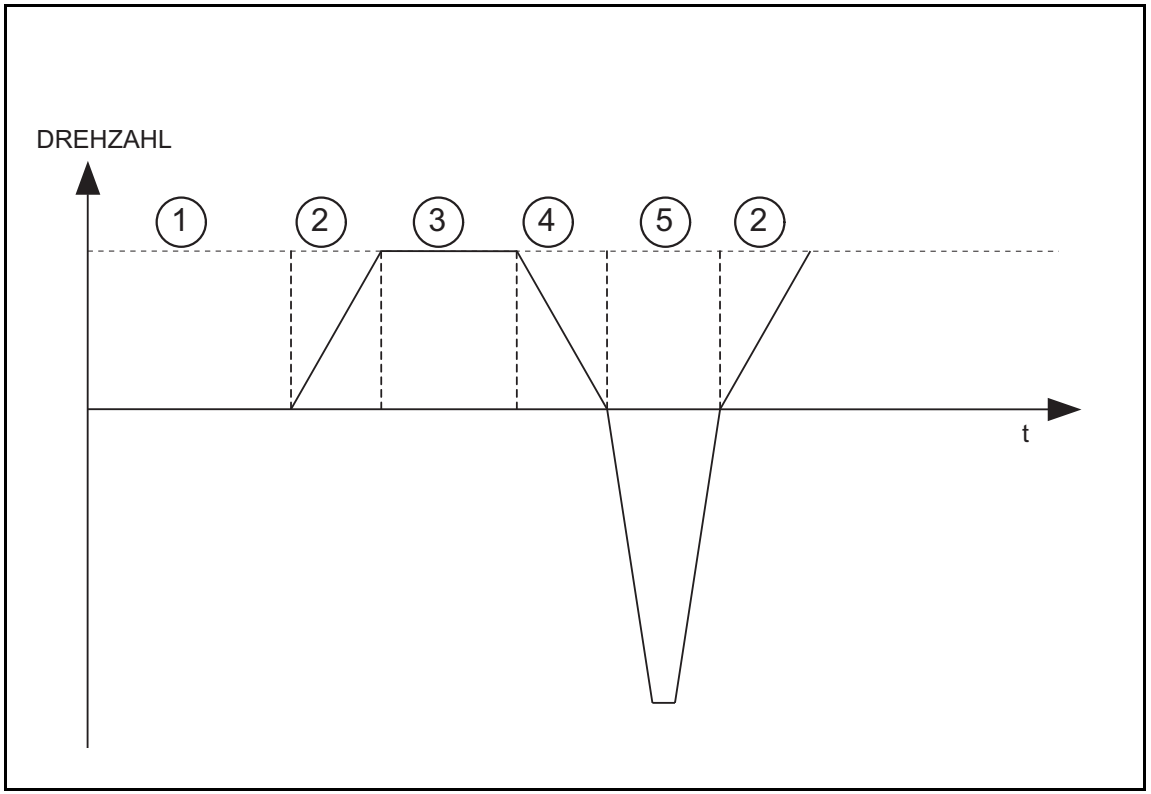
```
'Zuerst haben wir eine bestimmte Wartezeit, weil das
'Material gerade geschnitten worden ist
warte_strecke=schnitt_laenge-l_beschl/2
MOVELINK(0,warte_strecke,0,0,band_achse) AXIS(fliegende_achse)
WAIT UNTIL MTYPE AXIS(fliegende_achse)=22

'Das Band wird gestartet
FORWARD AXIS(band_achse)

schleife:

    'Bandgeschwindigkeit jeden Zyklus aktualisieren
    SPEED AXIS(band_achse)=band_geschwindigkeit

    'Schneidbewegung bei synchronisierter Drehzahl
    band_schnitt=synch_dist+l_beschl+l_verz
    schneidvorr_schnitt=synch_dist+l_beschl/2+l_verz/2
    MOVELINK(schneidvorr_schnitt,band_schnitt,l_beschl,l_verz,band_achse) AXIS(fliegende_achse)
    WAIT UNTIL MPOS AXIS(fliegende_achse)>l_beschl/2

    'Schneidwerkzeug aktivieren, wenn es in Synchronisation mit dem Band ist
    'Drehzahl zum Schnitt verringern
    SPEED AXIS(schneid_achse)=schnitt_drehzahl
    MOVEABS(end_pos) AXIS(schneid_achse)
    MOVEABS(0) AXIS(schneid_achse)
    WAIT UNTIL NTYPE AXIS(schneid_achse)=2
    'Schnelle Drehzahl für Rueckkehr
    WAIT LOADED AXIS(schneid_achse)
    SPEED AXIS(schneid_achse)=rueckkehr_drehzahl

    schnitt_zaehler=Schnitt_zaehler+1zoll

    'In Synchronisation mit dem Master so zurückkehren,
    'dass es keine Wartezeit gibt
    band_zurueck=Schnitt_laenge-synch_dist-l_verz-l_beschl
    schneidvorr_schnitt=l_beschl/2+synch_dist+l_verz/2)
    MOVELINK(-schneidvorr_schnitt,band_zurueck,l_beschl,/4,1_verz/4,band_achse) AXIS(fliegende_achse)

GOTO schleife
```

Der Drehzahl-Zeit-Graph zeigt die einzelnen Schritte des obigen Beispiels. Diese Schritte sind:

1. Der Anfangszyklus: Der Slave wartet auf die richtige Produktlänge für den Schnitt (schnitt_laenge – distanz_zu_beschleunigen/2). Es ist notwendig, distanz_zu_beschleunigen zu teilen, wenn wir den Befehl **MOVELINK** verwenden, denn wenn wir synchronisieren, legt der Master die doppelte Strecke des Slaves zurück.
2. Der Slave beschleunigt, um sich mit dem Master zu synchronisieren. Wenn die Beschleunigung beendet ist, beträgt die relative Distanz zwischen der Kante des Produkts und der Schneidvorrichtung schnitt_laenge.
3. Dies ist der Synchronisierungsteil: Die relative Distanz zwischen der Kante des Produkts und der Schneidvorrichtung bleibt gleich. Der Materialschnitt wird vorgenommen. Dadurch entsteht eine neue Materialkante.
4. Der Verzögerungsteil: Das Material fährt weiter, und die Schneidvorrichtung stoppt.
5. Zurückfahren mit hoher Drehzahl: Die Strecken sind wie folgt berechnet: Wenn der Slave seine Ausgangsposition erreicht, ist die Kante des Produkts in der richtigen Position, um einen neuen Schnitt zu starten.

Eine neue Bewegung beginnt (Schritt 2).

Abb.60



6.2.9 Korrekturprogramm

Diese Anwendung ist für einen Rotations-Etikettierer.
Die Konstanten sind:

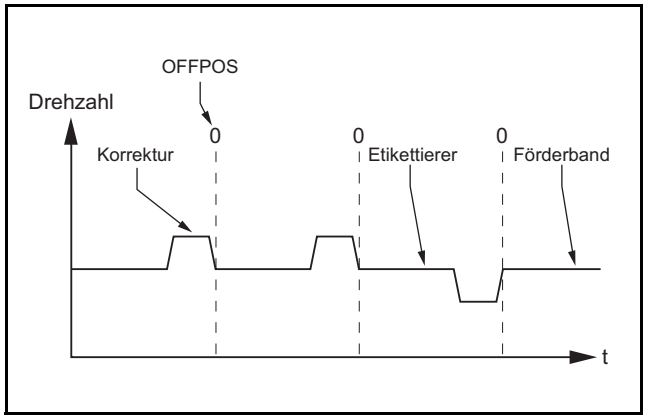
- Das Produkt kommt auf einem Förderband an (Master-Achse), das mit konstanter Geschwindigkeit läuft.
- Ein Rotations-Etikettierer, der 1:1 mit dem Förderband synchronisiert ist, bringt die Etiketten an.
- Der Abstand zwischen den Produkten ist fest und mechanisch garantiert.

Der Abstand zwischen den Etiketten ist niemals genau konstant, deshalb ist eine Korrektur erforderlich. Dies erfolgt durch Überordnung einer virtuellen Achse auf die Bewegung des Etikettierers.
Die Differenz zwischen der erwarteten Position und der tatsächlichen Position wird mit einer Fotozelle gemessen. Dies ist der Korrekturfaktor.
Jedesmal, wenn eine Korrektur vorgenommen wird, wird die Ausgangsposition entsprechend aktualisiert.

Beispiel

```
foerderband=0
etikettierer=1
virtuell=15
SERVO AXIS(foerderband)=1
SERVO AXIS(foerderband)=1
WDOG=1

BASE(etikettierer)
CONNECT(1, foerderband)
ADDAX(virtuell)
FORWARD AXIS(foerderband)
REGIST(1)
WAIT UNTIL MARK=0

schleife:
    WAIT UNTIL MARK
    Korrektur=REG_POS+erwartete_Pos
    MOVE(korrektur) AXIS(virtuell)
    WAIT IDLE AXIS(virtuell)
    OFFPOS=-etikett_laenge+korrektur
    REGIST(1)
    WAIT UNTIL MARK=0
GOTO schleife
```

Abb.61



7 Fehlersuche

7.1 Spannung und Analyse-Tools

Überprüfen Sie die Spannung der Stromversorgungs-Eingangsklemmen. Vergewissern Sie sich, dass die Spannung innerhalb des spezifizierten Bereichs liegt. Wenn die Spannung außerhalb des spezifizierten Bereichs liegt, funktioniert das System möglicherweise nicht richtig.

Um Fehler im TJ1-MC__ und TJ1-ML__ zu diagnostizieren und sie in diesen Baugruppen zu beheben, verwenden Sie das Trajexia Tools-Software-Tool. Um Fehler im TJ1-PRT zu diagnostizieren und sie in dieser Baugruppe zu beheben, verwenden Sie ein PROFIBUS-Konfigurations- und Überwachungs-Tool (z. B. **OMRON CX-PROFIBUS**).

Achtung
Klemmen Sie Kabel zunächst ab, bevor Sie prüfen, ob sie durchgebrannt sind. Selbst nach erfolgter Durchgangsprüfung besteht die Gefahr eines Stromschlags.

Achtung
Bei Verlust des Gebersignals kann der Servomotor durchlaufen, oder es kann eine Fehlfunktion auftreten. Achten Sie darauf, dass der Servomotor vom mechanischen System abgekoppelt ist, bevor Sie das Gebersignal prüfen.

Achtung
Achten Sie bei der Fehlersuche darauf, dass sich keine Personen im Bereich der Anlage aufhalten und dass auch beim Anlaufen des Servomotors keine Anlagenteile beschädigt werden können. Achten Sie darauf, dass die Apparatur über einen Notstopp sofort angehalten werden kann, falls der Servomotor anläuft.

7.2 TJ1-MC__

7.2.1 Systemfehler

Systemfehler werden im LED-Display des TJ1-MC__ als **Enn** angezeigt, wobei **nn** der Fehlercode ist.

Fehlercode	Beschreibung	Ursache	Lösung
E00	BASIC SRAM-Fehler	Hardware-Fehler des TJ1-MC__.	Tauschen Sie den TJ1-MC__ aus.
E01	System-SRAM-Fehler, niederwertiges Wort	Hardware-Fehler des TJ1-MC__.	Tauschen Sie den TJ1-MC__ aus.
E02	System-SRAM-Fehler, hochwertiges Wort	Hardware-Fehler des TJ1-MC__.	Tauschen Sie den TJ1-MC__ aus.
E03	Batterie-Ladezustandsfehler	Die Batteriespannung ist zu niedrig.	Tauschen Sie die Batterie aus.
...	Hardware-Fehler	Hardware-Fehler des TJ1-MC__.	Tauschen Sie den TJ1-MC__ aus.

Hinweis:
Weitere Informationen finden Sie in Abschnitt 3.2.254.

7.2.2 Achsenfehler

Achsenfehler werden im LED-Display des TJ1-MC__ als **Ann** angezeigt, wobei **nn** die Nummer der Achse ist, die den Fehler verursacht hat.
Es gibt zwei mögliche Ursachen:

- Der eingestellte Wert eines Achsenparameters ist falsch oder liegt außerhalb des Bereichs.
- Fehler oder Alarm bei dem Servotreiber, der der Achse zugeordnet ist.

Die zwei Ursachen mit ihren Lösungen sind wie folgt:

- Der Wert eines Achsenparameters ist falsch oder liegt außerhalb des Bereichs.
- Fehler oder Alarm bei dem Servotreiber, der der Achse zugeordnet ist

Der Wert eines Achsenparameters ist falsch oder liegt außerhalb des Bereichs

Wenn der Wert eines Achsenparameters falsch ist oder außerhalb des Bereichs liegt, tritt ein Achsenfehler auf. Auf dem Display des Servotreibers, der der Achse zugeordnet ist, wird kein Alarm oder Fehler angezeigt.
Sie können die Ursache des Fehlers mit dem Befehl **AXISSTATUS** anzeigen lassen. Geben Sie im Trajexia Tools-Terminalfenster **PRINT AXISSTATUS AXIS(nn)** ein, wobei **nn** die Achsenummer angibt. Der zurückgegebene Wert des Befehls **AXISSTATUS** enthält den Achsen-Fehlercode.
Siehe auch den Befehl **AXISSTATUS**.
Sie können auch das Fenster **Axis Parameter** (Achsenparameter) in Trajexia Tools öffnen und das Feld **AXISSTATUS** (Achsenstatus) der Achse überprüfen, die den Fehler verursacht hat. Die Bits, die die Ursache des Fehlers anzeigen, sind in roten Großbuchstaben dargestellt.
Um den Fehler zu beheben, führen Sie folgende Schritte durch:

1. Korrigieren Sie den Wert.
2. Setzen Sie den Controller zurück, oder klicken Sie auf die Schaltfläche **Axis status error** (Achsenstatus-Fehler).

Fehler oder Alarm bei dem Servotreiber, der der Achse zugeordnet ist

Wenn ein Fehler oder Alarm bei dem der Achse zugeordneten Servotreiber einen Achsenfehler verursacht, wird der Antriebsalarm auf dem LED-Display des Antriebs angezeigt. Sie können auch das Achsenparameter-Fenster in Trajexia Tools öffnen und das Feld **AXISSTATUS** (Achsenstatus) der Achse überprüfen, die den Fehler verursacht hat. Bei dem zurückgegebenen Wert des Befehls **AXISSTATUS** wird das zweite Bit (Bit **a**: Servotreiber-Kommunikationsfehler) und/oder das dritte Bit (Bit **m**: Servotreiber-Alarm) in roten Großbuchstaben angezeigt.
Um den Fehler zu beheben, führen Sie folgende Schritte durch:

1. Ziehen Sie das Servotreiber-Handbuch zu Rate, um die Ursache des Fehlers zu ermitteln, und beheben Sie den Fehler.
2. Setzen Sie den Controller zurück, oder klicken Sie auf die Schaltfläche **Axis status error** (Achsenstatus-Fehler).

7.2.3 Baugruppenfehler

Baugruppenfehler werden im LED-Display des TJ1-MC__ als **Unn** angezeigt, wobei **nn** die Nummer der Baugruppe ist, die den Fehler verursacht hat.
Es gibt vier mögliche Ursachen:

- Fehlerhafte Baugruppe.
- Die Baugruppe ist nicht an den Trajexia-Bus angeschlossen.
- Eine E/A-Baugruppe oder ein Frequenzumrichter an einer MECHATROLINK-II-Baugruppe ist ausgefallen oder nicht angeschlossen.
- Kein Abschlusswiderstand.

Fehlerhafte Baugruppe

Der Fehlercode **U0n** wird auf dem Display angezeigt, wobei **n** eine Zahl von 0 bis 6 ist und die Nummer der Baugruppe angibt, die den Fehler verursacht. Um das Problem zu lösen, tauschen Sie die fehlerhafte Baugruppe aus.

Die Baugruppe ist nicht an den Trajexia-Bus angeschlossen

Der Fehlercode **U0n** wird auf dem Display angezeigt, wobei **n** eine Zahl von 0 bis 6 ist und die Nummer der Baugruppe angibt, die den Fehler verursacht. Um das Problem zu lösen, überprüfen Sie die Bus-Steckverbindung der Baugruppe.

Eine E/A-Baugruppe oder ein Frequenzumrichter an einer MECHATROLINK-II-Baugruppe ist ausgefallen oder nicht angeschlossen

Der Fehlercode **U0n** wird auf dem Display angezeigt, wobei **n** die Nummer des TJ1-ML__ angibt, an den die MECHATROLINK-II-Baugruppe angeschlossen ist, die den Fehler verursacht.

Sie können System-Merker setzen, um diese Fehlermeldungen zu aktivieren und zu deaktivieren. Standardmäßig sind die Fehlermeldungen aktiviert.
Um die Fehlermeldungen zu deaktivieren, geben Sie **COORDINATOR_DATA(7,1)** im Trajexia Tools-Terminalfenster ein.

Um die Fehlermeldungen zu aktivieren, geben Sie **COORDINATOR_DATA(7,0)** im Trajexia Tools-Terminalfenster ein.

Um die aktuelle Einstellung anzeigen zu lassen, geben Sie **PRINT COORDINATOR_DATA(7)** im Trajexia Tools-Terminalfenster ein.

Um den Fehler nach der Reparatur zu löschen, führen Sie folgende Schritte durch:

- Schließen Sie die ausgefallene MECHATROLINK-II-E/A-Baugruppe oder den Frequenzumrichter wieder an.
- Geben Sie MECHATROLINK(n, 5, station, –1) im Trajexia Tools-Terminalfenster ein:
  Hierbei ist **n** die Nummer des TJ1-ML__, an den die betreffende MECHATROLINK-II-Baugruppe angeschlossen ist, und **Station** ist die Gerätenummer der ausgefallenen MECHATROLINK-II-Baugruppe.

Wenn Sie das System ohne das ausgefallene Gerät verwenden wollen, können Sie alle verfügbaren Geräte am TJ1-ML__ wieder anschließen. Geben Sie zu diesem Zweck **MECHATROLINK(n, 0)** im Trajexia Tools-Terminalfenster ein, wobei **n** die Nummer des TJ1-ML__ ist, der den Fehler meldet.

Kein Abschlusswiderstand

Der Fehlercode **U07** wird auf dem Display angezeigt.
Um das Problem zu lösen, überprüfen Sie den Abschlusswiderstand, oder tauschen Sie den Abschlusswiderstand aus, wenn er defekt ist.

7.2.4 Konfigurationsfehler

Konfigurationsfehler werden im LED-Display des TJ1-MC__ als **Cnn** angezeigt, wobei **nn** die Nummer der Baugruppe ist, die den Fehler verursacht hat.
Ursachen eines Konfigurationsfehlers sind:

- Das System hat zu viele Baugruppen des gleichen Typs, und die Regeln für das Hinzufügen von Baugruppen zu einem System wurden nicht eingehalten.
- Sie haben zu viele MECHATROLINK-II-Stationen an den TJ1-ML__ angeschlossen.
- Es gibt zu viele Achsen im System.
- Es gibt zu viele MECHATROLINK-II-Stationen im System, die nicht mit Achsen verknüpft sind.

Um dieses Problem zu lösen, ändern Sie das System so, dass die Regeln für das Hinzufügen von Baugruppen zu einem System eingehalten werden. Siehe Hardware-Referenzhandbuch.

7.2.5 Austauschen der Batterie

Um die Sicherungsbatterie auszutauschen, führen Sie folgende Schritte durch:

1. Stellen Sie sicher, dass die Spannungsversorgungsbaugruppe mindestens fünf Minuten lang eingeschaltet ist. Andernfalls ist der Kondensator, der den Speicher des TJ1-MC__ sichert, wenn die Batterie nicht angeschlossen ist, nicht vollständig aufgeladen, was zu Datenverlust im Speicher führen kann.
2. Ziehen Sie den Deckel des Batteriefachs von der Baugruppe weg, um das Batteriefach zu öffnen.
3. Ziehen Sie an den roten und weißen Drähten, um die alte Batterie herausnehmen zu können.
4. Stellen Sie sicher, dass Sie die nächsten zwei Schritte innerhalb von 30 Sekunden abgeschlossen haben, um Datenverlust im RAM-Speicher zu verhindern.
5. Klemmen Sie die Drähte von der alten Batterie ab.
6. Klemmen Sie die Drähte an der neuen Batterie an.
7. Setzen Sie die neue Batterie in das Batteriefach ein.
8. Schließen Sie die Abdeckung des Batteriefachs.

7.3 TJ1-PRT

7.3.1 Systemfehler

Anzeige	Problem	Lösung
Die LEDs leuchten oder blinken nicht	Die Spannungsversorgung ist ausgeschaltet.	Schalten Sie die Spannungsversorgung ein.
	Der TJ1-PRT ist defekt.	Tauschen Sie den TJ1-PRT aus.
Die ERH-LED leuchtet.	Kommunikationsausfall zwischen TJ1-MC__ und TJ1-PRT.	Setzen Sie den TJ1-MC__ zurück. Wenn dies nicht hilft, tauschen Sie den TJ1-MC__ aus.
Die ERC-LED leuchtet.	Baugruppenfehler. Der TJ1-PRT ist defekt.	Tauschen Sie den TJ1-PRT aus.

7.3.2 E/A-Datenübertragungsprobleme

Anzeige	Problem	Lösung
Die COMM-LED ist aus und die BF-LED leuchtet	Die PROFIBUS-Konfiguration ist falsch, es gibt keine Kommunikation mit dem Master.	• Überprüfen Sie, ob der TJ1-PRT die gleiche Stationsadresse wie in der Konfiguration des Masters hat. • Stellen Sie sicher, dass keine Stationsadresse zweimal benutzt wird.
	Die PROFIBUS-Verkabelung ist nicht korrekt.	• Überprüfen Sie, ob die richtigen Pole der CN1-Steckverbindung angeschlossen sind. • Überprüfen Sie, ob ein Kurzschluss oder eine Leitungsunterbrechung vorliegt. • Überprüfen Sie, ob der richtige Kabeltyp verwendet wurde. • Stellen Sie sicher, dass die Leitungsansätze nicht zu lang sind.
	Das PROFIBUS-Netzwerk ist nicht richtig angeschlossen.	Schließen Sie das PROFIBUS-Netzwerk an den richtigen Stellen an.
	Die PROFIBUS-Masterbaugruppe ist defekt.	Tauschen Sie die Masterbaugruppe aus.
	Der TJ1-PRT ist defekt.	Tauschen Sie den TJ1-PRT aus.
Die COMM-LED ist aus und die BF-LED blinkt	Die PROFIBUS-Konfiguration ist falsch, es gibt keine Kommunikation mit dem Master.	• Überprüfen Sie, ob die richtige GSD-Datei im Master verwendet wurde. • Überprüfen Sie die Konfigurations- und Parameterdaten des Slaves. • Überprüfen Sie, ob das Netzwerk zur Kommunikation mit der Baudrate konfiguriert ist, die vom TJ1-PRT unterstützt wird.
	Es sind keine Konfigurationsdaten für den Slave ausgewählt.	Überprüfen Sie die Konfiguration im Master.
	Der TJ1-PRT ist defekt.	Tauschen Sie den TJ1-PRT aus.

7.4 TJ1-DRT

7.4.1 Systemfehler

Anzeige	Problem	Lösung
Die LEDs leuchten oder blinken nicht	Die Spannungsversorgung ist ausgeschaltet.	Schalten Sie die Spannungsversorgung ein.
	Der TJ1-DRT ist defekt.	Tauschen Sie den TJ1-DRT aus.
Die ERH-LED leuchtet.	Kommunikationsausfall zwischen TJ1-MC__ und TJ1-DRT.	Setzen Sie den TJ1-MC__ zurück. Wenn dies nicht hilft, tauschen Sie den TJ1-MC__ aus.
Die ERC-LED leuchtet	Baugruppenfehler. Der TJ1-DRT ist defekt.	Tauschen Sie den TJ1-DRT aus.

7.4.2 E/A-Datenübertragungsprobleme

Anzeige	Problem	Lösung
Die NOK-LED blinkt und die NF-LED ist aus	Der DeviceNet-Master kommuniziert nicht mit dem TJ1-DRT.	• Konfigurieren und starten Sie den DeviceNet-Master.
Die NOK-LED ist aus und die NF-LED leuchtet	Teilnehmeradressen-Duplizierungsfehler.	• Überprüfen Sie die Teilnehmeradresse.
	Netzwerkkabelfehler.	• Überprüfen Sie die Netzwerkkabel.

7.5 TJ1-ML__

7.5.1 Systemfehler

Anzeige	Problem	Lösung
Alle LEDs sind aus	Die Spannungsversorgung ist ausgeschaltet.	Schalten Sie die Spannungsversorgung ein.
	Der TJ1-ML__ ist defekt.	Tauschen Sie den TJ1-ML__ aus.

7.5.2 Bus-Fehler

Anzeige	Problem	Lösung
Die BF-LED leuchtet	Kabelfehler im MECHATROLINK-II-Bus.	Überprüfen Sie die MECHATROLINK-II-Kabel zwischen den Stationen, die an die Baugruppe angeschlossen sind, auf Unterbrechungen und Unregelmäßigkeiten (Kurzschluss zwischen den Kommunikationsleitungen A und B, Kurzschluss in einer Kommunikationsleitung mit Abschirmung).
	Der MECHATROLINK-II-Bus-Abschlusswiderstand fehlt oder ist beschädigt.	Installieren Sie einen MECHATROLINK-II-Bus-Abschlusswiderstand an der letzten Station in der Kette, oder tauschen Sie ihn aus.
	Die an die Baugruppe angeschlossene MECHATROLINK-II-Station ist durch Abschaltung der Spannungsversorgung oder durch einen MECHATROLINK-II-Schnittstellenfehler an der Station ausgefallen.	Überprüfen Sie die Spannungsversorgung und die MECHATROLINK-II-Schnittstelle der Station, die das Problem verursacht hat. Tauschen Sie die Station nötigenfalls aus.
	Der TJ1-ML__ ist defekt.	Tauschen Sie den TJ1-ML__ aus.

Hinweis:
Nachdem die Ursache eines Fehlers beseitigt ist, muss der MECHATROLINK-II-Bus der Baugruppe, bei der der Fehler aufgetreten ist, neu initialisiert werden. Geben Sie Folgendes in das Trajexia Tools-Terminalfenster ein:
MECHATROLINK(n, 0)
Hierbei ist **n** die Nummer der Baugruppe, an die die Baugruppe, die den Fehler verursacht hat, angeschlossen ist.

7.6 TJ1-FL02

7.6.1 Systemfehler

Anzeige	Problem	Lösung
Alle LEDs sind aus	Die Spannungsversorgung ist ausgeschaltet.	Schalten Sie die Spannungsversorgung ein.
	Der TJ1-FL02 ist defekt.	Tauschen Sie den TJ1-FL02 aus.
Die RUN-LED leuchtet, die A EN- oder B EN-LED ist aus	Die Achse, bei der die EN-LED nicht leuchtet, ist nicht aktiviert.	Aktivieren Sie die Achse: Führen Sie den Befehl **WDOG=ON** und/ oder **AXIS_ENABLE** an der Achse aus.
Die RUN-LED leuchtet, die A EN- oder B EN-LED blinkt	Es liegt ein Achsenfehler an der Achse vor, bei der die EN-LED blinkt.	Der TJ1-MC__ zeigt die Nummer der Achse mit dem Achsenfehler an. Beheben Sie die Ursache des Achsenfehlers, und löschen Sie den Achsenfehler oder starten Sie das System neu.

A

Anwendungserstellung 191
Anwendungsfenster 194

B

BASIC
- Datenstrukturen 25
- Mathematische Spezifikationen 27
- Variablen 25

BASIC-Befehle 33
BASIC-Programme 30
BASIC-Programmierung 24
Batterie 302
Befehl
- Achse 33
- Programm 37
- Programmsteuerung 38
- System 38
- Task 40

befehl
- E/A 36
- Kommunikation 36

Befehlszeilen-Schnittstelle 30
Beispiel
- CAM-Tabelle 294
- Datenaufzeichnung und Überwachung 269
- Einachs-Programm 286
- Einstellung von Einheiten 239
- Geschwindigkeits-Betriebsart 230
- Initialisierungsprogramm 283
- Korrekturprogramm 298
- Nullpunktsuch 253
- Nullpunktsuche 253
- Position in einem Gitter 289

Position mit Produkterkennung 287
Positions-Betriebsart 235
Registrierung 259
Schneidvorrichtungsprogramm 295
Servotreiber-Eigenschaften 251
Shell-Programm 279
STARTUP-Programm 225
Tüten-Zuführprogramm 291
Verstärkungseinstellungen 229
Beispiel für CAM-Tabelle 294
Beispiel für Datenaufzeichnung und Überwachung 269
Beispiel für eine Achse 286
Beispiel für Einheiten 239
Beispiel für Geschwindigkeits-Betriebsart 230
Beispiel für Initialisierung 283
Beispiel für Korrektur 298
Beispiel für Nullpunktsuch 253
Beispiel für Position in einem Gitter 289
Beispiel für Position mit Produkterkennung 287
Beispiel für Positions-Betriebsart 235
Beispiel für Schneidvorrichtung 295
Beispiel für Servotreiber-Eigenschaften 251
Beispiel für Tüten-Zuführprogramm 291
Beispiel für Verstärkung 229
Beispiele für Nullpunktsuche 253
Benutzerdefiniertes Protokoll 165
Bewegungsausführung 28

D

DEVICENET
Einrichtung der Kommunikation 173
Fehler 304
Kommunikationsstatus 178
Schnittstelle 173

E

E/A-Status 216
Ethernet-Protokoll 153

F

Fehler
- Achse 300
- Baugruppe 301
- Konfiguration 302
- TJ1-MC__ 300

FINS-Slave-Protokoll 156
Flex Achsmodulachse
- Fehler 305

Funktion
- E/A 36
- Mathematischer 37
- System 38

H

Hardware-Übersicht 23
Host-Link
- Basic-Befehle 159
- Master-Protokoll 158

Host-Link-
- Slaveprotokoll 163

I

Installation der Software 181
Intelligente Antriebe 208

J

Jog 217

K

Konstanten 36

M

MECHATROLINK
- Fehler 304
- Protokoll 179

Menü
- Anzeige 224
- Hilfe 224
- Optionen 221
- Programm 203
- Projekt 196
- Steuerung 198
- Tools 205

Menübeschreibungen 196
Modifikator
- Slot 38

Multitasking 23

N

Netzwerkanschluss 187

O

Operand 37
- Mathematischer 37

Oszilloskop 210

P

Parameter
- Achse 34
- E/A 36
- Kommunikations- 36
- Slot 38
- System 39
- Task 40

PC
- Anschluss 186

Dezentrale Verbindung 155
Direktanschluss 154
Spezifikationen 180
Trajexia-Tools starten 187
PROFIBUS
Einrichtung der Kommunikation 167
Fehler 302
Kommunikationsstatus 172
PROFIBUS-Schnittstelle 167
Programmdatei vergleichen 192
Programmier-Tool 180
Projekt vergleichen 192
Protokoll
Benutzerdefiniert 165
DEVICENET 173
FINS-Client 158
FINS-SLAVE 156
Host-Link-Master 158
Host-Link-Slave 163
MECHATROLINK 179
PROFIBUS 167
Trajexia-Tools 156
Protokollübersicht 153

R

Registrierungsbeispiel 259

S

Schnittstelle
DEVICENET 173
Ethernet 153
MECHATROLINK 179
PROFIBUS 167
Seriell 158
Schnittstellenübersicht 153

Serielle Schnittstelle 158
Shell-Beispiel 279
Sicherheit, Betriebsumgebung 17
Sicherheit, Montage von Baugruppen 21
STARTUP-Beispiel 225
STARTUP-Programm
modifizieren 208
Symbole 195
Systemübersicht 22

T

TABLE-Speicher-Ansicht 219
Trajexia Tools-Protokoll 156
Trajexia vergleichen 192

V

VR-Speicher-Editor 219

W

Warnhinweise, Sicherheit 16

Revisionshistorie

Die Revisionsnummer des Handbuchs bildet das Ende der Katalognummer, die auf der vorderen Umschlagseite des Handbuchs angegeben ist.

Überarbeitungsstand	Datum	Änderungen
01	August 2006	Original
02	Oktober 2006	DeviceNet-Aktualisierung
03	Mai 2007	Aktualisiert durch TJ1-MC04 und TJ1-ML04. Verbesserte BASIC-Befehle, Programmierbeispiele und Tipps.